

Руководство пользователя IPS Development Kit



Оглавление

1	Введение	13
2	Написание модулей расширения.....	14
2.1	Серверная часть	14
2.2	Клиентская часть	15
2.3	Получение и регистрация служб	15
2.4	Пользовательская сессия.....	16
2.4.1	Получение сессии на клиенте и сервере	16
2.4.2	IUserSession	17
2.4.3	UserSession	20
3	Метаданные.....	21
3.1	Языковые варианты	21
3.1.1	IDBLanguageCollection	21
3.1.2	IDBLanguageType	22
3.1.3	IDBLanguage	23
3.2	Предметные области	24
3.2.1	IDBSubjectAreaCollection	24
3.2.2	IDBSubjectAreaType	25
3.2.3	IDBSubjectArea.....	26
3.3	Уровни продвижения	27
3.3.1	IDBLifecycleLevelCollection.....	27
3.3.2	IDBLifecycleLevelType	29
3.3.3	IDBLifecycleLevel	30
3.4	Группы атрибутов.....	30
3.4.1	IDBAttributesGroupCollection.....	30
3.4.2	IDBAttributesGroup	32
3.5	Типы атрибутов	33
3.5.1	AttributeTypeProperties	33
3.5.2	IDBAttributeTypeCollection	33
3.5.3	IDBAttributeType.....	35
3.6	Типы объектов	42
3.6.1	ObjectTypeProperties	42
3.6.2	IDBObjectTypeCollection	42
3.6.3	IDBObjectType	45
3.6.4	Attribute4ObjectTypeProperties	48
3.6.5	IDBAttribute4ObjectTypeCollection.....	48
3.6.6	IDBAttributeType4Object	50
3.7	Схемы жизненных циклов объектов.....	51
3.7.1	IDBLCSchemaCollection	51
3.7.2	IDBLCSchema	51
3.8	Этапы жизненного цикла объектов.....	52
3.8.1	IDBLifecycleStepCollection.....	52
3.8.2	IDBLifecycleStep	54
3.9	Типы связей.....	56

3.9.1	RelationTypeProperties	56
3.9.2	IDBRelationTypeCollection	57
3.9.3	Attribute4RelationTypeProperties	59
3.9.4	IDBAttribute4RelationTypeCollection	60
3.9.5	IDBAttributeType4Relation	61
3.9.6	IDBRelationType	62
3.10	Допустимые связи между объектами	63
3.10.1	RelationsApplicabilityProperties	63
3.10.2	IDBRelationsApplicabilityCollection	64
3.10.3	IDBRelationsApplicability	66
3.11	Дополнительный кэш метаданных (класс MetaDataHelper)	68
3.11.1	Пространство имён	68
3.11.2	Свойства класса	68
3.11.3	Методы класса	68
3.11.4	Получение информации о типах объектов	69
3.11.5	Получение информации о типах связей	71
3.11.6	Получение информации об иерархии типов объектов	72
3.11.7	Получение информации о типах связей по умолчанию	73
3.11.8	Получение информации о допустимых типах связей	74
3.11.9	Получение информации о применяемости	74
3.11.10	Получение информации о специальных типах объектов	76
3.11.11	Получение информации о специальных типах связей	77
3.11.12	Получение информации о типах атрибутов	78
3.11.13	Класс IMSObjectType – описание типа объекта	80
3.11.14	Класс IMSRelationType – описание типа связи	81
3.11.15	Класс IMSAttributeType – описание типа атрибута	81
3.11.16	Класс IMSApplicability – описание применяемости	83
4	Атрибуты объектов и связей	84
4.1	Базовые перечислители, структуры, классы	84
4.1.1	Перечислитель GetAttributeValuesModes	84
4.1.2	Структура AttributeDataTableValue	84
4.2	Интерфейсы для работы со списками атрибутов	85
4.2.1	IDBAttributable	85
4.2.2	IDBAttributeCollection	87
4.3	Работа с экземпляром атрибута	90
4.3.1	Класс AttributeValues	90
4.3.2	Интерфейс IDBAttribute	91
4.4	Особенности работы с различными типами данных	95
4.4.1	Строки (ftString)	95
4.4.2	Целые числа (ftInteger и ftAutoInc)	95
4.4.3	Вещественные числа (ftDouble)	96
4.4.4	Дата и время (ftDateTime)	96
4.4.5	Тексты произвольной длины (ftMemo)	96
4.4.6	Глобальные идентификаторы (ftGuid)	98

4.4.7	Ссылки на версию объектов (ftObjectLink)	98
4.4.8	Пароли (ftPassword)	99
4.4.9	Вещественные числа, выраженные в единицах измерения (ftMeasured)	100
4.4.10	Файлы и двоичные данные (ftFile, ftBlob, ftShortBlob)	104
4.4.11	Логический тип данных (ftBoolean)	108
5	Информационные объекты	110
5.1	Базовые перечислители, структуры, классы, интерфейсы	110
5.1.1	Перечислитель AttributeSourceTypes	110
5.1.2	Перечислитель ColumnContents	110
5.1.3	Перечислитель ColumnNameMapping	110
5.1.4	Перечислитель LogicalOperators	111
5.1.5	Перечислитель ObjectFiltrationState	111
5.1.6	Перечислитель ObligatoryObjectAttributes	112
5.1.7	Перечислитель RelationalOperators	113
5.1.8	Перечислитель SortOrders	114
5.1.9	Структура ColumnDescriptor	114
5.1.10	Структура ColumnInfo	115
5.1.11	Структура ConditionStructure	116
5.1.12	Структура DBRecordSetParams	117
5.1.13	Структура QuickObjectInfo	119
5.1.14	Класс ObligatoryObjectAttributesHelper	120
5.1.15	Класс QueryConsts	121
5.1.16	Интерфейс IDeletable	121
5.1.17	Интерфейс IDBGuid	122
5.1.18	Интерфейс IDBSessionable	122
5.1.19	Интерфейс IDBRecords	122
5.1.20	Структура DocumentTypeSettings	123
5.1.21	Интерфейс IDocumentTypeSettingsService	124
5.2	Права доступа к объектам	126
5.2.1	Перечислитель ActionCategory	126
5.2.2	Перечислитель AccessType	126
5.2.3	Перечислитель ActionType	126
5.2.4	Перечислитель CheckAccessFlags	127
5.2.5	Структура ActionProperties	127
5.2.6	Интерфейс IDBSecurity	127
5.2.7	Класс DBSessionable	130
5.3	Стандартные информационные объекты	130
5.3.1	Интерфейс IDBObject	131
5.3.2	Класс DBObject	136
5.4	Работа с проектами	137
5.4.1	Перечислитель ProjectFiltrationModes	137
5.4.2	Интерфейс IDBProjectObject	137
5.4.3	Класс ProjectParticipantInfo	139
5.4.4	Класс ProjectParticipantInfoEx	139

5.5	Создание и поиск объектов.....	139
5.5.1	Интерфейс IDBObjectCollection	139
5.5.2	Интерфейс IDBObjectService	142
5.5.3	Интерфейс ICreatorContainer	143
5.5.4	Интерфейс IDBObjectCreator	143
5.5.5	Интерфейс IDBObjectCollectionCreator	143
5.6	Примеры.....	144
5.6.1	Как создавать объекты базы данных.....	144
5.6.2	Как получить список объектов.....	144
5.6.3	Как создавать свои обработчики объектов определённого типа.....	146
5.6.4	Как подписаться на изменение значения атрибута объектов и связей	146
5.6.5	Создание обработчиков объектов своих типов.....	147
6	Связи между объектами.....	148
6.1	Связь между объектами	148
6.1.1	Интерфейс IDBRelation	148
6.1.2	Класс DBRelation	149
6.2	Создание и поиск связей.....	150
6.2.1	Интерфейс IDBRelationCollection	150
6.2.2	Интерфейс IDBRelationService	154
6.2.3	Интерфейс IDBRelationCreator	155
6.2.4	Интерфейс IDBRelationCollectionCreator.....	156
6.2.5	Интерфейс ICompositionLoadService	156
6.3	Фильтрация состава.....	158
6.3.1	Правила подбора версий	158
6.3.2	Варианты значений переменных в правилах подбора	162
6.3.3	Настройки фильтрации составов.....	163
6.3.4	Делегат FiltrationChanged.....	164
6.3.5	Перечислитель VersionRuleType	164
6.3.6	Интерфейс IFiltrationClass.....	164
6.3.7	Интерфейс IFiltrationSettings	165
6.3.8	Интерфейс IFiltrationService	166
6.3.9	Класс CompareFunctionsHelper	168
6.3.10	Класс ComparableValue	168
6.3.11	Класс VersionRuleCriterion.....	168
6.3.12	Класс VersionsRule	169
6.4	Примеры.....	169
6.4.1	Создание обработчиков коллекций связей определенных типов.....	169
7	Службы сервера приложений.....	171
7.1	Обработка событий	171
7.1.1	Интерфейс IEventLogHelper.....	171
7.1.2	Интерфейс IEventLog	177
7.1.3	Структура EventlogSettings	178
7.1.4	Примеры	179
7.2	Работа с выборками и классификаторами	179

7.2.1	Интерфейс ISelectionsService	179
7.3	Чтение и сохранение настроек	184
7.3.1	Интерфейс IDBConfigurations.....	184
7.3.2	Перечислитель DBConfigMode.....	191
7.3.3	Интерфейс IContainerService	191
7.3.4	Примеры	194
7.4	Вспомогательные службы.....	194
7.4.1	Интерфейс ICacheDataset	194
7.4.2	Класс Attribute4ID.....	197
7.4.3	Интерфейс IBrowseFolder	197
7.5	Анализ списка удаляемых объектов	198
7.5.1	Интерфейс IObjectsDeleteAnalyzerService	198
7.5.2	Класс DeletingObject	200
7.5.3	Класс DeletingObjects	202
7.5.4	Интерфейс IObjectsDeleteAnalyzer	205
7.5.5	Перечислитель DeleteAnalyzerJobProgress	205
7.5.6	Класс DeleteAnalyzerJobStatus	206
7.5.7	Класс DeletingObjectComparer	207
7.5.8	Класс DeletingObjectComparerCaption	207
7.5.9	Класс DeletingObjectComparerOwnerID	208
7.5.10	Класс DeletingObjectComparerChkOutBy	209
7.5.11	Класс DeletingObjectComparerObjectType	210
7.5.12	Класс DeletingObjectComparerLCStep	210
7.5.13	Класс DeletingObjectComparerBaseVersion.....	211
7.5.14	Класс DeleteAnalyzerJob.....	211
7.5.15	Класс ObjectsDeleteAnalyzerService	213
7.6	Удаление объектов	214
7.6.1	Интерфейс IObjectsDeleteService.....	214
7.6.2	Перечислитель DeleteObjectsJobMode	216
7.6.3	Перечислитель DeleteObjectsJobProgress.....	216
7.6.4	Класс DeleteObjectsJobStatus	216
7.6.5	Класс DeleteObjectsJob	219
7.6.6	Класс ObjectsDeleteService.....	220
7.7	Пакетное взятие объектов на изменение	222
7.7.1	Интерфейс IObjectsCheckOutServerService	222
7.7.2	Интерфейс IAssignable	223
7.7.3	Класс ObjectVersionDescription	223
7.7.4	Класс ObjectCheckOutVersionDescription	227
7.7.5	Перечислитель ObjectVersionDescriptionOptions	228
7.7.6	Перечислитель ObjectCheckedOutVersionMode	228
7.7.7	Класс ObjectVersionDescriptionsHelper	229
7.8	Индексация содержимого документов	230
7.8.1	Написание конвертера файлов	231
7.8.2	Регистрация конвертера файлов.....	232

7.8.3	Создание пользовательских процедур экспертной системы	233
8	Службы и компоненты универсального клиента	234
8.1	Создание объектов и их версий	234
8.1.1	IOBJECTCREATORSERVICE	234
8.1.2	Замена стандартного мастера создания объектов	236
8.2	Контекстное меню	238
8.2.1	Добавление команд в контекстное меню	238
8.2.2	Интерфейс ICommandsProvider	238
8.2.3	Класс CommandsInfo	239
8.2.4	Класс CommandInfo	240
8.2.5	Добавление элементов в шаблон контекстного меню	240
8.2.6	Стандартные команды контекстного меню	241
8.2.7	Перекрытие существующих команд в контекстном меню	241
8.2.8	Приоритеты команд контекстных меню – класс TriggerPriority	241
8.2.9	Дополнительная настройка команд меню	242
8.2.10	Класс AdjustableMenuCommand	242
8.2.11	Класс AdjustableMenuCommands	243
8.3	Редактирование атрибутов объектов и связей	244
8.3.1	Класс ObjectPropertyGrid	244
8.4	Диалоги и формы вывода информации	245
8.4.1	Диалог для выбора метаданных	245
8.4.2	Диалог для выбора объектов	246
8.4.3	Класс SelectionWindow	246
8.4.4	Перечислитель SelectionOptions	248
8.4.5	Перечислитель DynamicSelectionMode	249
8.4.6	Делегат DynamicSelectionEventHandler	249
8.4.7	Делегат SelectionWindowBeforeShow	250
8.4.8	Перечислитель SelectedItemsAnalyzerResult	250
8.4.9	Интерфейс ISelectedItemsAnalyzer	250
8.4.10	Класс ObjectTypesSelectedItemsAnalyzer	251
8.4.11	Интерфейс IBackgroundTaskView	251
8.4.12	Интерфейс IBackgroundTask	252
8.4.13	Перечислитель BackgroundTaskChangedType	253
8.4.14	Перечислитель BackgroundTaskState	254
8.4.15	Перечислитель BackgroundTaskShowMode	254
8.4.16	Делегат BackgroundTaskChangedEventHandler	254
8.4.17	Интерфейс IOutputView	255
8.4.18	Класс ProgressForm	255
8.4.19	Класс IMMessageBox	257
8.4.20	Класс IMMessageBoxButton	258
8.4.21	Перечислитель IMMessageBoxImage	259
8.4.22	Перечислитель MessageBoxButtonsAdv	259
8.5	Чтение и запись на сервер файлов и двоичных данных	259
8.5.1	Класс BlobProcReader	259

8.5.2	Класс BlobProcWriter.....	260
8.5.3	Перечислитель BlobProcessorMode.....	262
8.5.4	Примеры	262
8.6	Добавление пунктов в общее дерево настроек системы	263
8.6.1	Интерфейс IPropertyPagesService	263
8.6.2	Интерфейс IPropertyPage.....	263
8.6.3	Перечислитель PropertyPageType.....	264
8.7	Обработка исключений	264
8.7.1	Класс ExceptionForm	264
8.8	Буфер обмена (Clipboard).....	265
8.8.1	Интерфейс IClipboard	265
8.8.2	Класс ClipboardObject.....	266
8.8.3	Класс ClipboardObjectsList.....	267
8.8.4	Интерфейс ICutCopy.....	268
8.9	Вспомогательные классы и службы.....	268
8.9.1	Класс FormStorage	268
8.9.2	Регистрация расширений Навигатора – служба IFactory	269
8.9.3	Интерфейс ISimpleSelectedItems	271
8.9.4	Интерфейс ISelectedItems.....	272
8.10	Отображение статусов объектов и связей	272
8.10.1	Общие сведения.	273
8.10.2	Запрос статусов со стороны клиентского приложения.	273
8.10.3	Запрет добавления статусов некоторым серверным плагинам.....	273
8.10.4	Регистрация серверного плагина для обработки статусов.....	274
8.10.5	Обработка статусов на серверной стороне.	275
8.10.6	Получение списка зарегистрированных серверных плагинов.....	275
8.10.7	Получение статусов на клиентской стороне.	276
8.10.8	Получение описаний и изображений для статусов.	276
8.11	Передача дополнительных параметров в запросах Select.....	276
8.11.1	Служба IClientPluginsService – дополнительные параметры запросов.....	276
8.11.2	IClientPluginsDataTransfer – передача дополнительных параметров	278
8.12	Механизм отображения и сортировки составов в «Навигаторе»	278
8.12.2	Класс CompositionsAutosortRule.....	279
8.12.3	Класс ParentObjectType	282
8.12.4	Класс ChildRelationType.....	283
8.12.5	Класс ChildObjectType.....	286
8.12.6	Перечислитель RulesEditorAccessRights.....	287
8.12.7	Интерфейс ICurrentUserAndRole	287
8.12.8	Получение информации о видимых типах связей	288
8.12.9	Получение информации о порядке типов связей и объектов в составе.....	288
8.13	Создание закладок в «Навигаторе».....	288
8.13.1	Создание элемента управления с закладкой	288
8.13.2	Создание и регистрация провайдера для закладки.....	289
8.14	Работа со службой уведомлений	290

8.14.1	Интерфейс INotificationService	290
8.14.2	Делегат NotificationEventHandler	291
8.14.3	Класс NotificationEventNames.....	291
8.14.4	Класс NotificationEventArgs	292
8.14.5	Использование стандартных уведомлений	293
8.14.6	Создание собственного уведомления	293
8.14.7	Подписка на события службы уведомлений	294
8.15	Работа с менеджером команд	294
8.15.1	Интерфейс ICommandManager	295
8.15.2	Интерфейс ICommandState.....	296
8.15.3	Интерфейс ICommandTarget	297
8.15.4	Интерфейс ICommandTarget2	297
8.16	Панели инструментов универсального клиента	298
8.16.1	Класс BarManager	298
8.16.2	Класс ToolBarContainer	300
8.16.3	Класс ToolBar.....	301
8.16.4	Создание новых панелей инструментов	303
8.16.5	Поиск существующих панелей инструментов	304
8.17	Создание окон, поддерживающих докинг и сохранение состояния	304
8.17.2	Добавление возможности сохранять и восстанавливать состояние	304
8.18	Работа с диспетчером событий.....	304
8.18.1	Перечислитель IOEventFlags	305
8.18.2	Перечислитель IOEventType.....	305
8.18.3	Перечислитель IOEventTypes	305
8.18.4	Интерфейс IOEvent.....	306
8.18.5	Класс IOEvent.....	306
8.18.6	Интерфейс IODispatcher.....	306
8.18.7	Класс IODispatcher.....	307
8.18.8	Интерфейс IOSource.....	307
8.18.9	Интерфейс IODestination.....	308
8.19	Создание новых типов узлов для «Навигатора».....	308
8.19.1	Класс NodeColumn.....	309
8.19.2	Класс NodeColumnCollection	312
8.19.3	Перечислитель NodeColumnSortOrder	315
8.19.4	Перечислитель CellTransformationMode.....	315
8.19.5	Перечислитель ContentType.....	316
8.19.6	Перечислитель NodeQueryOptions	316
8.19.7	Интерфейс INodeColumnScheme	316
8.19.8	Интерфейс INodeColumnTransform.....	318
8.19.9	Класс CellValue	318
8.19.10	Интерфейс IDescriptor	319
8.19.11	Интерфейс INodeItems.....	320
8.19.12	Интерфейс INode	322
8.19.13	Интерфейс IContextAware.....	324

8.19.14	Интерфейс INodeID	324
8.19.15	Класс NodeIDPath	325
8.19.16	Класс PartSlot	326
8.19.17	Интерфейс INodePart	326
8.19.18	Интерфейс INodeQuerySupport	327
8.19.19	Интерфейс INodeQuery	328
8.19.20	Интерфейс IUpdateAnalyser	329
8.19.21	Интерфейс IUpdatePlan	330
8.20	Работа с экспертной системой	331
8.20.1	Перечислитель ExpertResult	331
8.20.2	Перечислитель ExpertTraceFlags	332
8.20.3	Класс CalcAttrPair	332
8.20.4	Класс CalculatedAttr	333
8.20.5	Интерфейс IExpertTask	333
8.20.6	Интерфейс IExpertUser	336
9	Структура файлов портфеля	337
9.1	Папки портфеля	337
9.2	Файлы портфеля	337
9.2.1	BriefcaseConfig.xml	337
9.2.2	ExportContent.xml	338
9.2.3	Metadata.xml	338
9.2.4	MetadataExportList.xml	338
9.2.5	Objects.xml	339
9.2.6	Relations.xml	340
9.2.7	ObjAttributes.xml, RelAttributes.xml	340
9.2.8	ObjLcSteps.xml	343
9.2.9	MetadataSecurity.xml, ObjSecurity.xml	343
9.2.10	Contexts.xml	344
9.3	Системные константы	344
9.3.1	Категории	344
9.3.2	Категории действий	345
9.3.3	Вид действия	345
10	Формы редактирования и элементы ввода и отображения информации	347
10.1.1	Интерфейс IFormDesignerControl	347
10.1.2	Класс FormDesignerFormLinksProviderType	347
10.1.3	Интерфейс IFormDesignerFormLinksManager	348
10.1.4	Класс FormLink	348
10.1.5	Интерфейс IFormDesignerFormLinksProvider	349
10.1.6	Интерфейс IFormDesignerActionParams	350
10.1.7	Класс FormDesignerAction	350
10.1.8	Интерфейс IFormDesignerActionHandler	350
10.1.9	Интерфейс IFormDesignerActionManager	351
10.1.10	Интерфейс IFormDesignerEventHandlerBase	352
10.1.11	Интерфейс IFormDesignerFormEventsHandler	352

10.1.12	Интерфейс IAttributeChangingEventHandler	352
10.1.13	Класс AttributeChangingEventArgs	353
10.1.14	Интерфейс IFormDesignerEventsManager	354
10.1.15	Класс AttributeInfo	355
10.1.16	Перечисление RefreshMode	355
10.1.17	Класс DesForm	355
10.1.18	Интерфейс IBaseDesForm.....	358
10.1.19	Интерфейс IAttributeEditorModified	358
10.1.20	Интерфейс IAttributeEditor.....	358
10.1.21	Перечисление AttributeDestinationPoint	359
10.1.22	Интерфейс IParent4Control.....	359
10.1.23	Интерфейс IValidateBeforeSave	360
10.1.24	Интерфейс IImageFromLibrary.....	360
10.1.25	Класс IMLabel	360
10.1.26	Класс IMPictureBox	360
10.1.27	Класс AttrCheckedListBox	361
10.1.28	Класс AttrComboBox	361
10.1.29	Класс AttrDateEdit.....	361
10.1.30	Интерфейс IIMControlEnabled	361
10.1.31	Класс IMGroupBox	361
10.1.32	Класс IMPanel	362
10.1.33	Класс IMTabControl.....	362
10.1.34	Класс AttrButton	362
10.1.35	Класс AttrsControl	362
10.1.36	Класс AttrCheckBox.....	363
10.1.37	Класс AttrLabel.....	363
10.1.38	Класс AttrTextEdit.....	363
10.1.39	Класс AttrMaskedTextEdit	364
10.1.40	Класс AttrMemoEdit.....	364
10.1.41	Класс AttrPassword	364
10.1.42	Класс AttrTextBtn.....	364
10.1.43	Класс AttrTextBtnComp	365
10.1.44	Класс AttrMeasuredEdit	365
10.1.45	Класс AttrListBoxBase	366
10.1.46	Класс AttrListBox	367
10.1.47	Класс AttrListBox	367
10.1.48	Класс AttrMeasuredListBox	367
10.1.49	Перечисление ListContext	367
10.1.50	Класс ObjectsList.....	367
10.2	Примеры.....	369
10.2.1	Создание пользовательского события для кнопки	369
11	Разработка сценариев на языке C#.....	371
11.1	Модель сценариев C#.....	371
11.2	Компиляция сценариев	371

11.3	Выполнение сценариев	371
11.4	Требования к коду сценариев C#	371
11.5	Проверка кода сценариев	372
11.6	Обязательные условия корректного выполнения сценариев	372
11.7	Отладка сценариев	372
11.8	Рекомендации при написании сценариев:	373
11.9	Примеры сценариев C#	373
11.9.1	Пример сценария Workflow.	373
11.9.2	Пример сценария типа «Сценарий для шагов ЖЦ».	374
11.9.3	Пример сценария «Сценарий для кнопок на формах ввода данных».	374
12	IPS SDK	376
12.1	Критические изменения	376
12.2	Системные требования	376
12.3	Разработка на C#	376
12.3.1	Подготовка окружения разработчика на C#	376
12.3.2	Шаблоны проектов на C#	377
12.3.3	Создание нового солюшена	377
12.3.4	Добавление проектов в солюшен	378
12.3.5	Конфигурирование IPS SDK для солюшена	378
12.3.6	Конфигурирование проектов в солюшене	379
12.3.7	Отладка проектов в солюшене	381
12.3.8	Особенности пользовательских WebAPI-модулей	382
12.3.9	Просмотр списка доступных эндпоинтов и документации по WebAPI через Swagger UI ...	382
12.3.10	Взаимодействие с WebApi через Swagger UI	383
12.3.11	Адаптация модулей расширения прежних версий IPS под новое IPS SDK	385
12.4	Разработка на Typescript	385
12.4.1	Подготовка окружения разработчика на Typescript	385
12.4.2	Шаблоны проектов на Typescript	386
12.4.3	npm-пакеты для IPS SDK	386
12.4.4	Создание клиентского модуля расширения для IPS WebInterface	387
12.4.5	Отладка	397
	Приложение 1 Типы данных атрибутов IPS Development Kit	400
	Приложение 2 Стандартные команды контекстных меню	401

1 Введение

IPS Development Kit представляет собой универсальную систему управления информационными объектами. IPS Development Kit функционирует под управлением Microsoft.NET Framework и построен по трехзвенной архитектуре Client/Server. Для хранения и обработки данных используются СУБД MS SQL Server, Oracle и PostgreSQL. Непосредственно с СУБД работает сервер приложений Intermech Application Server посредством технологии ADO.NET, а клиентская часть IPS Development Kit подключается к серверу приложений через .NET Remoting.

И сервер приложений, и клиент IPS Development Kit позволяют загружать модули расширения, которые могут добавлять и модифицировать функциональность системы. При этом модули расширения могут наследовать свои классы и интерфейсы от общих классов и интерфейсов IPS Development Kit, подписываться на различные события, возникающие в системе, использовать функциональность различных служб IPS Development Kit, добавлять и модифицировать клиентские меню и панели инструментов и т.д.

Данное руководство предназначено для программистов, которые будут разрабатывать модули расширения к системе IPS Development Kit, программы для интеграции IPS Development Kit с различными информационными системами предприятия, конвертеры для импорта баз данных в IPS Development Kit и т.п. Все примеры в данном руководстве приведены на языке C#, однако архитектурные возможности Microsoft.NET Framework позволяют разрабатывать модули расширения на любом .NET-совместимом языке программирования, а также вызывать функции динамических библиотек и приложений Windows.

В комплект поставки системы входят примеры модулей расширения серверной и клиентской части IPS Development Kit с исходными текстами на языке C# (подкаталог Examples в каталоге с серверной установкой системы). В тексте руководства приведены ссылки на данные примеры.

Внимание! Запрещается производить модификацию данных непосредственно в СУБД в обход IPS Development Kit API! Это может привести к нарушению целостности или полной потере информации, хранящейся в базе данных!

Примечание: в руководстве могут быть описаны не все общие классы, интерфейсы и функции модулей, входящих в комплект поставки IPS Development Kit. При возникновении вопросов с функционированием каких-либо методов и интерфейсов рекомендуем обращаться в службу технической поддержки Интермех.

2 Написание модулей расширения

Универсальный клиент и сервер приложений разработаны таким образом, чтобы поддерживать архитектуру с расширяемой функциональностью. Суть этой архитектуры состоит в том, что клиент и сервер представляют собой набор базовых сервисов, обеспечивающих подключаемые модули средствами для интеграции с ядром.

2.1 Серверная часть

Загружаемый модуль представляет собой обычную сборку .NET. Для того чтобы загрузчик ядра мог выполнить инициализацию расширения, в этой сборке должны быть реализованы классы (один или несколько), наследуемые от интерфейса `Intermech.Interfaces.Plugins.IPackage`. Этот интерфейс требует наличия двух методов:

```
void Load(IServiceProvider serviceProvider);  
void Unload();
```

Загрузка и инициализация модуля расширения выполняется в следующей последовательности:

1. После загрузки сборки в память средствами .NET загрузчик получает список типов классов этой сборки (`public` или `internal`) и проверяет, наследованы ли они от интерфейса `Intermech.Interfaces.Plugins.IPackage`.
2. Если да, то создается экземпляр этого типа. При этом вызывается конструктор типа без параметров.
3. Созданный экземпляр приводится к интерфейсу `IPackage` и вызывается метод `Load` этого интерфейса. В качестве аргумента в метод передается служба управления сервисами клиента или сервера. Эта служба поддерживает интерфейс `IServiceProvider`, у которого загружаемый модуль может запросить все остальные службы через метод `GetService(typeof(тип_службы))`.
4. Дополнительно проверяется, поддерживает ли экземпляр расширения интерфейс `IConfigurable`. Если поддерживает, то вызывается метод `LoadConfiguration(IConfigurationManager configurationManager)` для того, чтобы модуль расширения смог прочитать свои настройки. При завершении приложения (для клиента) вызывается метод `SaveConfiguration(IConfigurationManager configurationManager)`, что позволяет сохранить настройки модуля расширения и прочитать их при последующем запуске.
5. Эта процедура выполняется для всех типов, описанных в п.1.

Ниже представлен скелет модуля расширения.

```
internal class PluginSample : IPackage, IConfigurable  
{  
    #region IPackage Members  
        public void Unload()  
        {  
        }  
        public string Name  
        {  
            get  
            {  
                return "Пример модуля расширения";  
            }  
        }  
        public void Load(IServiceProvider serviceProvider)  
        {  
        }  
    #endregion  
  
    #region IConfigurable Members
```

```

public void SaveConfiguration(IConfigurationManager configurationManager)
{
    IConfiguration cfg = configurationManager.Create("PluginSample");
    cfg.SetProperty("Prop", "PropValue");
}

public void LoadConfiguration(IConfigurationManager configurationManager)
{
    IConfiguration cfg = configurationManager.Open("PluginSample");
    if( cfg != null)
    {
        string str = cfg.GetProperty("Prop");
    }
}

#endregion
}

```

Метод Unload() не вызывается и оставлен для совместимости с предыдущими версиями системы.

Список загружаемых на сервере модулей расширения хранится в файле с именем сервера приложений и расширением *.cfg. При использовании ConsoleServer файл конфигурации называется ConsoleServer.exe.cfg

Секция в файле конфигурации выглядит следующим образом:

```

<PluginManager>
    <Plugin Location="PluginSample.dll" />
</PluginManager>

```

2.2 Клиентская часть

Загрузка клиентских плагинов осуществляется двумя способами. Основной способ – назначение загрузки модуля для роли, в которой пользователь авторизуется в клиенте IPS Development Kit. Загрузку клиентского плагина можно также назначить группам пользователей, включив эти группы в состав соответствующего объекта типа **Загружаемый модуль**. На конкретном компьютере загрузить клиентский модуль можно через меню **Настройка/Модули Расширения/Загрузить модуль расширения**.

Клиентский плагин должен поддерживать тот же набор интерфейсов, что и серверный. Метод Unload() не вызывается и оставлен для совместимости с предыдущими версиями системы.

Примечание: следует отметить, что некоторые классы, реализованные в сборке Intermech.Interfaces, требуют предварительной инициализации. Универсальный клиент и сервер приложений при старте производят их инициализацию, поэтому для клиентских и серверных модулей расширения ее выполнять не нужно.

2.3 Получение и регистрация служб

Для предоставления модулям расширения различной функциональности ядро системы на клиенте и сервере регистрирует так называемые службы, интерфейсы которых описаны в общих интерфейсных сборках и могут быть получены через статические классы Intermech.Interfaces.Server.ServerServices (на сервере приложений) и Intermech.Interfaces.Client.ServicesManager (на клиенте). Модули расширения могут регистрировать собственные службы для предоставления своей функциональности другим модулям системы. Ниже приведен пример регистрации собственной службы на сервере и получения ее интерфейса.

```

using Intermech.Interfaces.Server;
public interface IMyService
{
    // Методы интерфейса
}
private class MyService: IMyService
{
    // Реализация методов интерфейса IMyService
}
...
// Метод регистрации службы MyService
private void RegisterMyService()

```

```

{
    MyService service = new MyService();
    ServerServices.AddService(typeof(IMyService), service);
}
//Пример получения интерфейса службы
IMyService service = ServerServices.GetService(typeof(IMyService)) as IMyService;
Для регистрации на сервере службы, интерфейс которой можно было бы получать на клиенте,
предназначена служба ICustomServices. Ниже приведен пример регистрации серверной службы MyService для
работы с ней в клиентской части системы. Естественно, интерфейс службы IMyService в таком случае должен
быть описан в сборке, которая доступна как на стороне сервере, так и на стороне клиента.
// Регистрация серверной службы
MyService service = new MyService();
ICustomServices custServices = ServerServices.GetService(typeof(ICustomServices)) as ICustomServices;
custServices.AddService(typeof(IMyService), service);
// Получение серверной службы на клиенте
using (SessionKeeper keeper = new SessionKeeper())
{
    IUserSession session = keeper.Session;
    IMyService mService = session.GetCustomService(typeof(IMyService)) as IMyService;
}

```

2.4 Пользовательская сессия

Все действия в системе выполняются от имени авторизованной пользовательской сессии (интерфейс `Intermech.Interfaces.IUserSession`), что позволяет производить проверку прав доступа и аудит выполнения действий, а также выполнять модификацию данных в рамках транзакций с возможностью отката проведённых изменений. Большинство серверных интерфейсов, работающих с данными, получают с помощью методов пользовательской сессии.

2.4.1 Получение сессии на клиенте и сервере

Для серверных и клиентских приложений способы получения сессии различны. Причем на стороне сервера доступен не только интерфейс сессии, но и сам класс `Intermech.Kernel.UserSession`, реализующий функциональность сессии. Помимо методов, опубликованных в интерфейсе `IUserSession`, этот класс содержит ряд дополнительных функций (например, функции управления транзакциями).

2.4.1.1 Модули расширения сервера приложений

1. Получить ссылку на сессию можно у экземпляра любого класса, унаследованного от `DBSessionable`.

К таким классам относятся обработчики метаданных и списков метаданных, обработчики информационных объектов, связей и их списков и пр. Например,

```

using Intermech.Kernel;
class MyObject: DBObject
{
    public MyMethod()
    {
        UserSession mySession = this.UserSession;
        ...
    }
}

```

2. Если имеется локальный или глобальный идентификатор сессии, то с их помощью можно получить ссылку на сессию у статического класса `UserSession`. Этот метод используется для передачи сессии с клиента на сервер, поскольку клиент не может передавать полученный с сервера интерфейс `IUserSession` обратно на сервер. Пример:

```

using Intermech.Kernel;
public void ServerMethod(Guid sessionGuid)
{
    UserSession session = UserSession.GetSessionByID(sessionGUID) as UserSession;
}

```



```
...  
}
```

3. Для выполнения действий от имени системы на сервере приложений можно получить системную сессию. Это может понадобиться в случаях, когда авторизация пользователя невозможна: в момент инициализации модулей расширения, при работе по таймеру и пр. Пример:

```
using Intermech.Interfaces.Server;  
using Intermech.Interfaces;  
...  
IDBTimedEvents timedEvents = ServerServices.GetService(typeof(IDBTimedEvents)) as IDBTimedEvents;  
IUserSession sys_session = timedEvents.GetSystemSessionTemporaryClone("MyMethodName");  
try  
{  
    // Выполняем действия с системной сессией  
}  
finally  
{  
    // Освобождаем системную сессию  
    sys_session.Logout("MyMethodName");  
}
```

В метод получения клона системной сессии нужно передать строковый идентификатор этой сессии, который затем потребуется в методе отключения сессии Logout(). Это сделано для того, чтобы чужой код не смог по ошибке вызвать отключение сессии, которую создал ваш код, т.к. для отключения сессии методом Logout() требуется знать ее идентификатор.

Если нужно получить системную сессию, которая должна жить всё время существования сервера приложений, то такую сессию нужно получать методом GetSystemSessionPermanentClone.

4. Сессию можно клонировать с помощью метода IUserSession.Clone(string sessionName) (например, для выполнения фоновых операций в другом потоке). После завершения работы у клонированной сессии нужно вызвать метод Logout(string sessionName). Эта пара методов также требует идентификатор клона для его защиты от отключения сторонним кодом.

2.4.1.2 Модули расширения универсального клиента

Сессия получается с помощью вспомогательного класса SessionKeeper, который возвращает ссылку на сессию для текущего потока. Пример:

```
using Intermech.Interfaces.Client;  
using Intermech.Interfaces;  
...  
using (SessionKeeper keeper = new SessionKeeper())  
{  
    IUserSession session = keeper.Session;  
    ...  
}
```

Полученную таким образом сессию можно использовать только в рамках секции using класса SessionKeeper. Ее нельзя никуда передавать за пределами этой секции. Класс SessionKeeper обеспечивает правильное выделение сессии для текущего потока и корректное ее освобождение в пул сессий. Вне времени жизни класса SessionKeeper полученную у него сессию использовать нельзя.

2.4.2 IUserSession

Интерфейс пользовательской сессии. Содержит информацию о пользователе, авторизовавшемся в системе, а также функции для получения прочих интерфейсов, работающих с данными. Если сессия получена посредством вызова IMServer.Session, то перед вызовом свойств и методов интерфейса необходимо произвести авторизацию пользователя методом Login().

Пространство имен: Intermech.Interfaces

Примечание: описание функций получения различных интерфейсов см. в описании соответствующих интерфейсов.

2.4.2.1 Свойства интерфейса

AreaID

Строка с идентификаторами предметных областей, с которыми работает данная сессия. Инициализация строки производится в функции Login() и зависит от роли, в которой пользователь авторизовался в системе. Если строка пустая, то разрешен доступ к объектам из всех предметных областей.

Синтаксис:

```
string AreaID { get; set; }
```

ComputerName

Сетевое имя компьютера, с которого производилась авторизация сессии. Возвращает пустую строку в случае, если сессия не авторизована функцией Login().

Синтаксис:

```
string ComputerName { get; }
```

DeveloperMode

Если значение свойства установлено в *true*, то система загружена в режиме разработчика приложений. Этот режим позволяет, например, изменять глобальные идентификаторы уже созданных объектов и метаданных, что может понадобиться в процессе настройки базы данных и отладки модулей расширения системы. Включение режима разработчика производится в файле конфигураций (см. Руководство администратора, раздел ПЗ.1. Файл основных настроек сервера приложений, ключ DeveloperMode).

Синтаксис:

```
bool DeveloperMode { get; }
```

IsAdmin

Свойство возвращает значение *true*, если пользователь авторизовался в роли администратора системы.

Синтаксис:

```
bool IsAdmin { get; }
```

LanguageID

Идентификаторы языков, с которыми работает данная сессия. Если строка пустая, то сессия работает с языковым вариантом, установленным в системе по умолчанию.

Синтаксис:

```
string LanguageID { get; set; }
```

LastCallTime

Дата и время последнего обращения к функциям интерфейса (в глобальном времени). Используется системой для определения устаревших сессий, для которых по каким-либо причинам не была выполнена функция Logout().

Синтаксис:

```
DateTime LastCallTime { get; }
```

MaxRows

Свойство содержит количество строк, возвращаемых функциями пакетного чтения данных по умолчанию (например, при использовании константы QueryConsts.Default в параметрах DBRecordSetParams).

Синтаксис:

```
int MaxRows { get; set; }
```

RoleID

Идентификатор роли, в которой данный пользователь авторизовался в системе.

Синтаксис:

```
Int64 RoleID { get; }
```

SessionGUID

Глобальный идентификатор данной сессии. Свойство может использоваться для передачи идентификатора сессии с клиента в серверную службу, которая затем может получить интерфейс сессии с помощью статического метода UserSession.GetSessionByID().

Синтаксис:

```
Guid SessionGUID { get; }
```

ShowDeletedObjects

Свойство определяет, показывать ли в списках информационные объекты, находящиеся на уровне продвижения **Удалено**. Установка в *true* возможна только для сессии, авторизованной в роли **Администратор**.

Синтаксис:

```
bool ShowDeletedObjects { get; set; }
```

ShowPersonalObjects

Свойство определяет, показывать ли в списках информационные объекты, находящиеся на уровне продвижения **Персональный объект**. Установка в *true* возможна только для сессии, авторизованной в роли **Администратор**.

Синтаксис:

```
bool ShowPersonalObjects { get; set; }
```

TimeZoneOffset

Смещение времени временной зоны рабочей станции пользователя относительно универсального времени по Гринвичу (указывается в момент авторизации пользователя функцией Login()). Используется системой для пересчета локального времени в глобальное и наоборот.

Синтаксис:

```
TimeSpan TimeZoneOffset { get; }
```

UserID

Идентификатор подключенного пользователя (соответствует идентификатору IDBObject.ObjectID информационного объекта, который описывает данного пользователя в системе). Возвращает 0 в случае, если сессия не авторизована функцией Login().

Синтаксис:

```
Int64 UserID { get; }
```

UserName

Выводимое имя пользователя, авторизовавшегося в системе (только для чтения). До авторизации свойство содержит пустую строку.

Синтаксис:

```
string UserName { get; }
```

2.4.2.2 Методы интерфейса

Clone

Метод создает авторизованную копию текущей сессии (например, для выполнения фоновых операций) и возвращает ее интерфейс.

Синтаксис:

```
IUserSession Clone(string sessionName);
```

Параметры:

sessionName

Идентификатор клона. Предназначен для защиты клона от уничтожения кодом, который его не создавал.

GetCustomService

Метод позволяет получить на клиенте интерфейс серверной службы. Служба должна быть зарегистрирована на сервере в менеджере служб ICustomServices.

Синтаксис:

```
object GetCustomService(Type serviceType);
```

Параметры:

serviceType

Тип интерфейса службы, которую нужно получить.

Logout

Метод завершает работу пользователя в системе и освобождает занятые сессией ресурсы.

Синтаксис:

```
int Logout(string sessionName);
```

Параметры:

sessionName

Идентификатор клона, который был использован при его получении методом Clone().

Примечание: после завершения работы с копией сессии необходимо вызвать у нее метод Logout() для освобождения занимаемых ею ресурсов.

2.4.3 UserSession

Данный класс, реализующий функциональность пользовательской сессии, доступен только на стороне сервера. У серверных объектов, работающих только в контексте пользовательской сессии, данный класс доступен в виде свойства UserSession. Ниже приведено описание его методов, которые не опубликованы в интерфейсе IUserSession.

2.4.3.1 Свойства класса

InTransaction

Свойство возвращает true, если в текущей сессии стартована транзакция.

Синтаксис:

```
bool InTransaction { get; }
```

2.4.3.2 Методы класса

Commit

Метод завершает транзакцию, начатую методом StartTransaction, с сохранением всех изменений в базе данных.

Синтаксис:

```
void Commit();
```

Rollback

Метод отменяет все изменения, произведенные текущей сессией в базе данных, сделанные с момента начала текущей транзакции.

Синтаксис:

```
void Rollback();
```

StartTransaction

Метод начинает транзакцию, в рамках которой все изменения данных будут или приняты одновременно методом Commit, или отменены методом Rollback. Допускается вызов метода при уже стартованной транзакции. В этом случае сессия увеличит счетчик начатых транзакций, а все изменения будут записаны в базу только после вызова Commit для транзакции, стартовавшей раньше всех.

Синтаксис:

```
void StartTransaction();
```

Внимание! Открытую транзакцию необходимо заканчивать методами Commit или Rollback, иначе могут быть потеряны все изменения, сделанные в данной сессии после начала вашей транзакции! Ниже приведен пример правильного использования транзакции:

```
UserSession.StartTransaction();
try
{
    ... // Команды изменения данных
    UserSession.Commit();
}
catch
{
    UserSession.Rollback();
    throw;
}
```

3 Метаданные

Метаданными называют информацию, определяющую структуру и допустимые взаимосвязи хранимых в системе информационных объектов, а также способы их представления и модификации. К метаданным относятся: типы атрибутов, объектов и связей, группы атрибутов, уровни продвижения и схемы жизненных циклов, предметные области, языковые варианты.

Полные копии метаданных всегда хранятся в оперативной памяти сервера приложений и универсального клиента, поэтому доступ к ним на чтение не требует сетевых обращений (кроме первоначальной проверки прав доступа). Это значит, что операции чтения метаданных выполняются очень быстро, но общее количество метаданных в системе не должно быть слишком большим – не рекомендуется создавать в системе более ста тысяч типов атрибутов, объектов и связей.

Все интерфейсы обработчиков коллекций метаданных содержат метод `Select`, который возвращает таблицу со списком соответствующих метаданных. Поля этих таблиц описаны для каждого метода `Select` в следующем формате:

1. Имя поля в базе данных. Всегда приведено к верхнему регистру.
2. Тип данных поля в СУБД. Приведён в нотации MS SQL Server.
3. Комментарий, описывающий содержимое поля.
4. Имя соответствующего свойства в интерфейсе, предназначенном для редактирования метаданных этого типа. Если свойство содержится в другом интерфейсе, то перед именем свойства приведено имя этого интерфейса – в этом случае редактирующий интерфейс должен быть сначала приведен к данному интерфейсу, например:

```
IDBSubjectAreaType area = session.GetSubjectAreaType(myArea);  
Guid myAreaGuid = (area as IDBGuid).GUID;
```

Следует отметить, что все коллекции метаданных поддерживают интерфейс `IDBCollection`, который также содержит метод `Select`, возвращающий соответствующую таблицу со списком метаданных.

3.1 Языковые варианты

Языковым вариантом называют язык общения, на котором в данной информационной системе можно создавать атрибуты и записывать в них информацию.

Таблица в базе данных, в которой содержится список языковых вариантов: `IMS_LANGUAGES`

Пространство имен: `Intermech.Interfaces`

Интерфейс для работы со списком языковых вариантов: [IDBLanguageCollection](#).

Интерфейсы для работы с языковым вариантом: [IDBLanguageType](#) [IDBLanguage](#).

3.1.1 IDBLanguageCollection

Интерфейс позволяет создавать новые языковые варианты, получать список уже существующих языковых вариантов, проверять существование языкового варианта с указанным идентификатором и пр.

3.1.1.1 Способы получения интерфейса

[IUserSession.GetLanguageCollection\(\)](#)

3.1.1.2 Свойства интерфейса

[DefaultLanguageID](#)

Свойство возвращает идентификатор языкового варианта, который назначен в системе по умолчанию.

Синтаксис:

```
string DefaultLanguageID { get; }
```

3.1.1.3 Методы интерфейса

[CheckValidLanguageID](#)

Функция проверяет строку идентификаторов языковых вариантов и генерирует исключение `InvalidLanguageIDException` в случае, если строка содержит символы, не являющиеся идентификаторами языковых вариантов, содержащихся в базе данных системы.

Синтаксис:

```
void CheckValidLanguageID (string languageIDs);
```

Параметры:

languageIDs

Строка с идентификаторами языковых вариантов. Идентификаторы должны идти подряд без разделителей, например **ADB**.

Create

Метод создает новый языковой вариант.

Синтаксис:

```
char Create (string languageName, Guid guid, string cultureID)
```

Параметры:

cultureID

Идентификатор локализации (например, en – англоязычная локализация).

languageName

Уникальное наименование языкового варианта. Может содержать до 255 символов и не может принимать пустых значений.

guid

Уникальный глобальный идентификатор языкового варианта. Если в качестве параметра отправить Guid.Empty, то сервер сам сгенерирует глобальный идентификатор для создаваемого языкового варианта.

Возвращаемое значение:

Символьный идентификатор созданного языкового варианта.

Select

Метод возвращает таблицу со списком всех языковых вариантов, зарегистрированных в системе.

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка “F_LANGUAGE_NAME DESC” отсортирует список языковых вариантов по их наименованию в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_LANGUAGES.

addInfo

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица со списком языковых вариантов.

Таблица IMS_LANGUAGES. Список языковых вариантов.

Имя поля	Тип поля	Комментарий	IDBLanguageType
F_LANGUAGE_ID	CHAR(1)	Идентификатор языкового варианта.	LanguageID
F_LANGUAGE_NAME	NVARCHAR(255)	Наименование языкового варианта.	LanguageName
F_DEFAULT	SMALLINT	Является ли языковым вариантом, используемым по умолчанию (1 – да, 0 – нет).	IsDefaultLanguage
F_GUID	UNIQUEIDENTIFIER	Глобальный идентификатор языкового варианта.	GUID

3.1.2 IDBLanguageType

Данный интерфейс используется для модификации свойств языковых вариантов, а также для удаления языковых вариантов из системы. Реализация осуществляется объектом DBLanguageType.

3.1.2.1 Способы получения интерфейса

```
IUserSession.GetLanguage (string aLanguageID);
```

Параметры:

aLanguageID

Уникальный символ-идентификатор языкового варианта. Если в системе не зарегистрирован языковой вариант с таким идентификатором, то функция генерирует исключение KernelExceptionID номер 216.

```
IUserSession.GetLanguage (Guid guid);
```

Параметры:

guid

Глобальный идентификатор языкового варианта. Если в системе не зарегистрирован языковой вариант с таким глобальным идентификатором, то функция генерирует исключение `KernelExceptionID` номер 184.

3.1.2.2 Свойства интерфейса

GUID

Свойство позволяет прочитать или изменить глобальный идентификатор языкового варианта.

Синтаксис:

```
Guid GUID { get; set; }
```

IsDefaultLanguage

Свойство определяет, используется ли язык в системе по умолчанию.

Синтаксис:

```
bool IsDefaultLanguage { get; set; }
```

LanguageID

Свойство возвращает уникальный идентификатор языка.

Синтаксис:

```
string LanguageID { get; }
```

LanguageName

Свойство позволяет прочитать или модифицировать наименование языкового варианта.

Синтаксис:

```
string LanguageName { get; set; }
```

3.1.2.3 Методы интерфейса

Delete

Метод удаляет языковой вариант из системы. Генерирует исключение `KernelException` в случае, если данный языковой вариант где-либо используется.

Синтаксис:

```
int Delete(Int64 DeleteMode);
```

Параметры:

DeleteMode

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

3.1.3 IDBLanguage

Данный интерфейс используется для получения информации о языковом варианте, присвоенном атрибуту, типу атрибутов или группе атрибутов, а также для назначения языкового варианта типу или группе атрибутов. Реализация осуществляется соответственно объектами `DBAttribute`, `DBAttributeType` и `DBAttributesGroup`.

3.1.3.1 Способы получения интерфейса

Чтобы получить информацию о языковом варианте, назначенном атрибуту или группе атрибутов, необходимо привести реализующий объект или его интерфейс к интерфейсу `IDBLanguage`, например:

```
IDBLanguage language = (IDBAttributeType as IDBLanguage);
```

3.1.3.2 Свойства интерфейса

IsDefaultLanguage

Свойство возвращает значение `true`, если реализующему объекту назначен языковой вариант, используемый в системе по умолчанию (или языковой вариант вообще не назначен).

Синтаксис:

```
bool IsDefaultLanguage { get; }
```

LanguageID

Свойство возвращает и устанавливает символьный идентификатор языкового варианта, к которому относится данный атрибут или группа атрибутов. Пустая строка означает, что данный атрибут или группа относятся к языковому варианту, используемому в системе по умолчанию.

Синтаксис:

```
string LanguageID { get; set; }
```

LanguageName

Свойство возвращает наименование языка, который назначен реализующему объекту (атрибуту или группе атрибутов). Если языковой вариант реализующему объекту не назначен, то свойство возвращает пустую строку.

Синтаксис:

```
string LanguageName { get; }
```

3.2 Предметные области

Предметной областью называют категорию, позволяющую разделить всю хранящуюся в системе информацию на области применения. Это разделение позволяет упростить пользовательский интерфейс системы, исключив из списков типы объектов, связей и атрибутов, не относящиеся к предметным областям, с которыми работает данный пользователь.

Таблица в базе данных, в которой содержится список предметных областей: IMS_SUBJECT_AREAS

Пространство имен: Intermech.Interfaces.

Интерфейс для работы со списком предметных областей: IDBSubjectAreaCollection.

Интерфейсы для работы с предметной областью: IDBSubjectAreaType, IDBSubjectArea.

3.2.1 IDBSubjectAreaCollection

Интерфейс позволяет создавать новые предметные области, получать список уже существующих предметных областей, проверять существование предметных областей с указанными идентификаторами и пр. Реализация осуществляется объектом DBSubjectAreaCollection.

3.2.1.1 Способы получения интерфейса

IUserSession.GetSubjectAreaCollection()

3.2.1.2 Методы интерфейса

Create

Метод создает новую предметную область.

Синтаксис:

```
char Create (string areaName, string areaNote, Guid guid);
```

Параметры:

areaName

Уникальное наименование предметной области. Может содержать до 255 символов и не может принимать пустых значений.

areaNote

Описание предметной области (до 450 символов).

guid

Уникальный глобальный идентификатор предметной области. Если в качестве параметра отправить Guid.Empty, то сервер сам сгенерирует глобальный идентификатор для создаваемой предметной области.

Возвращаемое значение:

Символьный идентификатор созданной предметной области.

GetAreasCaption

Возвращает строку с наименованиями предметных областей, идентификаторы которых переданы в параметре areas.

Синтаксис:

```
string GetAreasCaption ( string areas);
```

Параметр:

areas

Идентификаторы предметных областей. Идентификаторы должны идти подряд без разделителей, например DBH.

Возвращаемое значение:

Наименования предметных областей, разделенные запятой (например, **Администрирование системы, Документооборот, Бухгалтерия**). Если параметр areas содержит пустую строку, то возвращается строка **Все**.

GetValidAreaID

Позволяет получить строку идентификаторов предметных областей, из которой будут отфильтрованы идентификаторы, не зарегистрированные в системе.

Синтаксис:

```
string GetValidAreaID (string anAreaID);
```

Параметр:

anAreaID

Исходная строка идентификаторов предметных областей. Идентификаторы должны идти подряд без разделителей, например “DBH”.

Возвращаемое значение:

Строка идентификаторов предметных областей, присутствующих в системе.

Select

Метод возвращает таблицу со списком всех предметных областей, зарегистрированных в системе.

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка F_AREA_NAME DESC отсортирует список предметных областей по их наименованию в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_SUBJECT_AREAS.

addInfo

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица со списком предметных областей.

Таблица IMS_SUBJECT_AREAS. Список предметных областей.

Имя поля	Тип поля	Комментарий	IDBSubjectAreaType
F_AREA_ID	CHAR(1)	Символ-идентификатор предметной области.	AreaID
F_AREA_NAME	NVARCHAR(255)	Наименование предметной области.	AreaName
F_AREA_NOTE	NVARCHAR(450)	Примечание для предметной области.	Note
F_GUID	VARCHAR(40)	Глобальный идентификатор предметной области.	IDBGuid.GUID

CheckValidAreaID

Функция проверяет, содержатся ли в базе предметные области с заданными в параметре anAreaID идентификаторами. Генерирует исключение InvalidAreaIDException в случае, если строка anAreaID содержит символы, не являющиеся идентификаторами предметных областей из базы данных системы.

Синтаксис:

```
void CheckValidAreaID (string anAreaID);
```

Параметр:

anAreaID

Строка идентификаторов предметных областей.

3.2.2 IDBSubjectAreaType

Данный интерфейс используется для модификации свойств предметных областей, а также удаления предметных областей из системы. Реализация осуществляется объектом DBSubjectAreaType.

3.2.2.1 Способы получения интерфейса

Для получения интерфейса используются функции:

```
IUserSession.GetSubjectAreaType (char aSubjectAreaTypeID)
```

Параметры:

aSubjectAreaTypeID

Символ-идентификатор предметной области.

IUserSession.GetSubjectAreaType (Guid guid)

Параметры:

guid

Глобальный идентификатор предметной области.

3.2.2.2 Свойства интерфейса

AreaID

Свойство возвращает символ-идентификатор предметной области.

Синтаксис:

```
char AreaID { get; }
```

AreaName

Свойство возвращает или модифицирует наименование предметной области.

Синтаксис:

```
string AreaName { get; set; }
```

Note

Свойство возвращает или модифицирует описание предметной области.

Синтаксис:

```
string Note { get; set; }
```

3.2.2.3 Методы интерфейса

Delete

Метод удаляет предметную область. Генерирует исключение `KernelException` в случае, если предметная область где-либо используется.

Синтаксис:

```
int Delete (Int64 DeleteMode);
```

Параметры:

М

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

SetGUID

Метод присваивает предметной области новый глобальный идентификатор.

Синтаксис:

```
void SetGUID ( Guid guid )
```

Параметры:

guid

Уникальный глобальный идентификатор предметной области.

3.2.3 IDBSubjectArea

Данный интерфейс используется для чтения и назначения предметных областей типам атрибутов, объектов и связей, группам атрибутов и уровням продвижения. Реализация осуществляется объектами `DBAttribute`, `DBAttributeType`, `DBObjectType`, `DBRelationType`, `DBAttributesGroup` и `DBLifecycleLevel`.

3.2.3.1 Способы получения интерфейса

Чтобы получить информацию о предметных областях, назначенных типу, группе атрибутов или уровню продвижения, необходимо привести интерфейс реализующего объекта к интерфейсу `IDBSubjectArea`, например:

```
IDBSubjectArea areas = UserSession.GetObjectType(objectTypeID) as IDBSubjectArea;
```

3.2.3.2 Свойства интерфейса

SubjectAreas

Свойство позволяет читать и записывать строку символов-идентификаторов предметных областей, к которым относится реализующий объект. Если строка пустая, то объект относится ко всем предметным областям.

Синтаксис:

```
string SubjectAreas { get; set; }
```

SubjectAreasCaption

Возвращает строку с наименованиями предметных областей, к которым относится данный объект. Например, **Машиностроение, Строительство**. Если реализующему объекту не присвоено ни одной предметной области, то свойство вернет строку "Все предметные области".

Синтаксис:

```
string SubjectAreasCaption { get; }
```

3.2.3.3 Пример использования

Данный пример получает список типов объектов верхнего уровня и выводит на системную консоль наименования предметных областей, назначенных каждому типу:

```
using System;
using System.Data;
using Intermech.Interfaces;
...
public static void testExample()
{
    IUserSession session;
    ... // тут код получения сессии (зависит от типа приложения)
    //получаем список типов объектов ObjectTypeListEx
    IDbObjectValueCollection ObjectTypeListEx;
    ObjectTypeListEx = session.GetObjectValueCollection(-1);
    DataTable tbEx;
    tbEx = ObjectTypeListEx.Select(Consts.F_OBJ_TYPE_NAME);
    //для каждого из типов получаем наименования предметных областей, назначенных типу
    foreach (DataRow row in tbEx.Rows)
    {
        IDbObjectType objType = session.GetObjectType(Convert.ToInt32(row[Consts.F_OBJECT_TYPE]));
        IDbSubjectArea area = objType as IDbSubjectArea;
        System.Console.WriteLine(String.Format("Тип объектов '{0}' относится к предметным областям '{1}'",
            objType.ObjectTypeName, area.SubjectAreasCaption));
    }
}
```

3.3 Уровни продвижения

Уровень продвижения объекта – это категория, определяющая статус объекта в процессе его жизненного цикла. Уровни продвижения служат для объединения одинаковых по смыслу шагов в жизненных циклах объектов различных типов, что позволяет унифицировать их обработку различными модулями системы.

Таблица в базе данных, в которой содержится список уровней продвижения: **IMS_LEVELS**.

Пространство имен: **Intermech.Interfaces**.

Интерфейс для работы со списком уровней продвижения: **IDBLifecycleLevelCollection**.

Интерфейсы для работы с уровнем продвижения: **IDBLifecycleLevelType**, **IDBLifecycleLevel**.

3.3.1 IDBLifecycleLevelCollection

Интерфейс позволяет создавать новые уровни продвижения, получать список уже существующих уровней продвижения и пр. Реализация осуществляется объектом **DBLifecycleLevelCollection**.

3.3.1.1 Способы получения интерфейса

Для получения интерфейса используются функции:

```
IUserSession.GetLifecycleLevelCollection ();
```

Возвращает интерфейс для работы с полным списком уровней продвижения.

```
IUserSession.GetLifecycleLevelCollection (bool filterRecs);
```

Возвращает интерфейс для работы с полным или фильтрованным списком уровней продвижения.

Параметры:

filterRecs

Если значение параметра filterRecs равно *true*, из списка исключаются уровни продвижения, которые данный пользователь не может видеть.

3.3.1.2 Методы интерфейса

Create

Метод создает новый уровень продвижения и возвращает его идентификатор.

Синтаксис:

```
int Create (string levelName, string litera, string areaID, Guid newGuid, bool isDefault);
```

Параметры:

levelName

Уникальное наименование уровня продвижения. Может содержать до 255 символов и не может принимать пустых значений.

litera

Литера уровня продвижения. Используется для сокращенного обозначения уровня продвижения (например, в штампах документации).

areaID

Строка с идентификаторами предметных областей, к которым будет относиться создаваемый уровень продвижения. Пустая строка означает, что уровень продвижения будет относиться ко всем предметным областям.

newGuid

Уникальный глобальный идентификатор уровня продвижения. Если в качестве параметра отправить Guid.Empty, то сервер сам сгенерирует глобальный идентификатор для создаваемого уровня продвижения.

isDefault

Если параметр isDefault = 1, то данный уровень продвижения будет использоваться для создания первого шага жизненного цикла объектов (с учетом предметной области типов объектов).

Возвращаемое значение:

Идентификатор созданного уровня продвижения.

Select

Метод возвращает таблицу со списком всех уровней продвижения, зарегистрированных в системе (или видимых данному пользователю, если получена фильтрованная коллекция уровней продвижения).

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка **F_LEVEL_NAME DESC** отсортирует список уровней продвижения по их наименованию в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_LEVELS.

addInfo

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица со списком уровней продвижения.

Таблица IMS_LEVELS. Список уровней продвижения.

Имя поля	Тип поля	Комментарий	IDBLifecycleLevelType
F_LEVEL_ID	INTEGER	Идентификатор уровня продвижения.	LevelID
F_LEVEL_NAME	NVARCHAR(255)	Наименование уровня продвижения.	LevelName
F_LITERA	NVARCHAR(32)	Литера уровня продвижения.	Litera
F_ICON	IMAGE	Иконка для изображения уровня продвижения.	LevelIcon
F_AREA_ID	NVARCHAR(8)	Предметные области, к которым относится уровень продвижения.	IDBSubjectArea. SubjectAreas
F_GUID	VARCHAR(40)	Глобальный идентификатор уровня продвижения.	GUID

F_DEFAULT	SMALLINT	Определяет, будет ли данный уровень использоваться для создания первого шага в схеме жизненного цикла объекта.	isDefault

3.3.2 IDBLifecycleLevelType

Данный интерфейс используется для модификации свойств уровней продвижения, а также удаления уровней продвижения из системы. Реализация осуществляется объектом DBLifecycleLevel.

3.3.2.1 Способы получения интерфейса

Для получения интерфейса используются функции:

```
IUserSession.GetLifecycleLevel (string levelName);
```

Параметры:

levelName

Уникальное наименование уровня продвижения. Функция генерирует исключение KernelExceptionID номер 173 в случае, если в системе не зарегистрирован уровень продвижения с указанным наименованием.

```
IUserSession. GetLifecycleLevel (int aLevelID)
```

Параметры:

aLevelID

Идентификатор уровня продвижения. Функция генерирует исключение KernelExceptionID номер 215 в случае, если в системе не зарегистрирован уровень продвижения с указанным идентификатором.

```
IUserSession.GetLifecycleLevel (Guid levelGuid);
```

Параметры:

levelGuid

Глобальный идентификатор уровня продвижения. Функция генерирует исключение KernelExceptionID номер 207 в случае, если в системе не зарегистрирован уровень продвижения с указанным глобальным идентификатором.

3.3.2.2 Методы интерфейса

Delete

Удаляет уровень продвижения.

Синтаксис:

```
int Delete (Int64 DeleteMode);
```

Параметры:

DeleteMode

Параметр зарезервирован для будущего использования. В случае каких-либо ошибок удаления метод генерирует исключения.

3.3.2.3 Свойства интерфейса

GUID

Глобальный идентификатор уровня продвижения.

Синтаксис:

```
Guid GUID { get; set; }
```

IsDefaultLevel

Определяет, будет ли данный уровень использоваться для создания первого шага в схемах жизненных циклов объектов.

Синтаксис:

```
bool IsDefaultLevel { get; set; }
```

LevelIcon

Иконка, отображающая уровень продвижения.

Синтаксис:

```
byte[] LevelIcon { get;set; }
```

LevelID

Идентификатор уровня продвижения.

Синтаксис:

```
int LevelID { get; }
```

LevelName

Уникальное наименование уровня продвижения. Может содержать до 255 символов и не может принимать пустых значений.

Синтаксис:

```
int LevelName { get; set; }
```

Litera

Литера уровня продвижения (например, 'A').

Синтаксис:

```
string Litera { get; set; }
```

3.3.3 IDBLifecycleLevel

Данный интерфейс используется для определения уровней продвижения объектов, шагов жизненного и типов атрибутов, а также для назначения уровня продвижения типам атрибутов. Реализация осуществляется объектами DBLifecycleStep, DBObject, DBAttributeType.

3.3.3.1 Способы получения интерфейса

Чтобы получить информацию об уровне продвижения, необходимо привести интерфейс реализующего объекта к интерфейсу IDBLifecycleLevel, например:

```
IDBLifecycleLevel lcLevel = UserSession.GetAttributeType(attributeID) as IDBLifecycleLevel;
```

3.3.3.2 Свойства интерфейса

LevelIcon

Иконка, отображающая уровень продвижения.

Синтаксис:

```
byte[] LevelIcon { get; }
```

LevelID

Идентификатор уровня продвижения.

Синтаксис:

```
int LevelID { get; set; }
```

LevelName

Наименование уровня продвижения.

Синтаксис:

```
int LevelName { get; }
```

Litera

Литера уровня продвижения.

Синтаксис:

```
string Litera { get; }
```

3.4 Группы атрибутов

Все типы атрибутов в системе делятся на смысловые группы, что облегчает поиск и обработку атрибутов, а также упрощает назначение прав доступа к атрибутам. Любой тип атрибута обязательно входит в системную группу **Все атрибуты**, а также может входить в произвольное количество других групп атрибутов. Строго говоря, группы атрибутов содержат именно типы атрибутов, но визуальнo эта группировка распространяется и на экземпляры атрибутов, принадлежащие объектам и связям.

Таблица в базе данных, в которой содержится список групп атрибутов: IMS_ATTR_GROUPS.

Пространство имен: Intermech.Interfaces.

Интерфейс для работы со списком групп атрибутов: IDBAttributesGroupCollection.

Интерфейс для работы с группой атрибутов: IDBAttributesGroup.

3.4.1 IDBAttributesGroupCollection

Интерфейс позволяет создавать новые группы атрибутов, а также получать список уже существующих групп атрибутов. Реализация осуществляется объектом DBAttributesGroupCollection.

3.4.1.1 Способы получения интерфейса

Для получения интерфейса используются функции:

```
IUserSession.GetAttributesGroupCollection ();
IUserSession.GetAttributesGroupCollection (bool filterRecs);
```

Параметры:

filterRecs

Если значение параметра filterRecs равно *true*, из списка исключаются группы атрибутов, которые пользователь не может видеть.

3.4.1.2 Методы интерфейса

Create

Метод создает новую группу атрибутов и возвращает её идентификатор.

Синтаксис:

```
int Create (string groupName, string groupNote, string languageID, string areaID, Guid guid);
```

Параметры:

groupName

Уникальное наименование группы атрибутов. Может содержать до 255 символов и не может принимать пустых значений.

groupNote

Комментарий к группе атрибутов, может содержать до 450 символов.

languageID

Идентификатор языкового варианта, к которому относится создаваемая группа атрибутов.

areaID

Строка с идентификаторами предметных областей, к которым будет относиться создаваемая группа атрибутов. Пустая строка означает, что группа будет относиться ко всем предметным областям.

guid

Уникальный глобальный идентификатор группы атрибутов. Если в качестве параметра отправить Guid.Empty, то сервер сам сгенерирует глобальный идентификатор для создаваемой группы атрибутов.

Возвращаемое значение:

Идентификатор созданной группы атрибутов.

Select

Метод возвращает таблицу со списком всех групп атрибутов, зарегистрированных в системе.

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка **F_GROUP_NAME DESC** отсортирует список групп атрибутов по их наименованию в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_ATTR_GROUPS.

addInfo

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица со списком групп атрибутов.

Таблица IMS_ATTR_GROUPS. Список групп атрибутов.

Имя поля	Тип поля	Комментарий	IDBAttributesGroup
F_GROUP_ID	INTEGER	Идентификатор группы атрибутов.	GroupID
F_GROUP_NAME	NVARCHAR(255)	Наименование группы атрибутов.	GroupName
F_NOTE	NVARCHAR(450)	Комментарии.	Note
F_AREA_ID	NVARCHAR(8)	Идентификаторы предметных областей, к которым относится данная группа атрибутов.	IDBSubjectArea. SubjectAreas
F_LANGUAGE_ID	CHAR(1)	Идентификатор языкового варианта.	IDBLanguage. LanguageID

F_GUID	VARCHAR(40)	Глобальный идентификатор группы атрибутов.	IDBGuid.GUID

3.4.2 IDBAttributesGroup

Интерфейс объекта, управляющего группой атрибутов.

3.4.2.1 Способы получения интерфейса

Для получения интерфейса используются функции:

```
IUserSession.GetAttributesGroup (int aGroupID);
```

Параметры:

aGroupID

Идентификатор группы атрибутов. Функция генерирует исключение KernelExceptionID номер 217 в случае, если в системе не зарегистрирована группа атрибутов с указанным идентификатором.

```
IUserSession.GetAttributesGroup (Guid guid);
```

Параметры:

guid

Глобальный идентификатор группы атрибутов. Функция генерирует исключение KernelExceptionID номер 185 в случае, если в системе не зарегистрирована группа атрибутов с указанным глобальным идентификатором.

3.4.2.2 Свойства интерфейса

Attributes

Коллекция типов атрибутов, входящих в состав данной группы.

Синтаксис:

```
IDBAttributeTypeCollection Attributes { get; }
```

GroupID

Идентификатор группы.

Синтаксис:

```
int GroupID { get; }
```

GroupName

Уникальное наименование группы атрибутов. Может содержать до 250 символов и не может быть пустым.

Синтаксис:

```
string GroupName { get; set; }
```

Note

Комментарии к группе атрибутов (до 450 символов).

Синтаксис:

```
string Note { get; set; }
```

3.4.2.3 Методы интерфейса

Delete

Метод удаляет группу атрибутов.

Синтаксис:

```
int Delete (Int64 DeleteMode);
```

Параметры:

DeleteMode

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

ExcludeAttribute

Метод исключает атрибуты из группы атрибутов.

Синтаксис:

```
int ExcludeAttribute (params int[] attributeIDs);
```

Параметры:

attributeIDs

Массив с идентификаторами типов атрибутов, которых нужно исключить из состава данной группы.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

IncludeAttribute

Метод включает тип атрибута в состав данной группы.

Синтаксис:

```
int IncludeAttribute (int attributeID);
```

Параметры:

attributeID

Идентификатор типа атрибута, включаемого в группу. Если данный тип уже включен в составе группы, то генерируется исключение KernelExceptionID номер 218.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

IncludeAttribute

Метод включает перечисленные типы атрибутов в состав данной группы атрибутов.

Синтаксис:

```
int IncludeAttribute (int[] attributeIDs);
```

Параметры:

attributeIDs

Массив с идентификаторами типов атрибутов, которых нужно включить в состав данной группы. Если в составе группы уже есть хотя бы один из указанных атрибутов, то генерируется исключение KernelExceptionID номер 218.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

SetGUID

Метод присваивает группе новый глобальный идентификатор.

Синтаксис:

```
void SetGUID (Guid guid).
```

Параметры:

guid

Уникальный глобальный идентификатор группы атрибутов.

3.5 Типы атрибутов

Типом атрибута называется описатель именованной характеристики, которая может быть присвоена объекту или связи того или иного типа (например, **Наименование**, **Масса**, **Фамилия** и т.п.). Тип атрибута определяет тип, размерность и способы ввода данных, которые можно присваивать экземпляру атрибута, необходимость хранения истории введенных значений, фильтрацию атрибута в списках атрибутов и пр.

Пространство имен: Intermech.Interfaces.

Интерфейс для работы со списком типов атрибутов: IDBAttributeTypeCollection.

Интерфейс для работы с типом атрибута: IDBAttributeType.

3.5.1 AttributeTypeProperties

Структура, позволяющая описать все свойства типа атрибута. Поля структуры соответствуют свойствам интерфейса IDBAttributeType. Кроме основного конструктора, в который нужно передать значения всех полей структуры, имеются также дополнительные конструкторы, позволяющие инициализировать структуру значениями по умолчанию, строкой таблицы с описателем типа атрибута IMS_ATTRIBUTES, а также значениями из другой структуры AttributeTypeProperties.

3.5.2 IDBAttributeTypeCollection

Интерфейс предназначен для регистрации в системе новых типов атрибутов, а также для получения списка атрибутов, находящихся в указанной группе атрибутов. Реализация осуществляется объектом IDBAttributeTypeCollection.

3.5.2.1 Способы получения интерфейса

ISession.GetAttributeTypeCollection (int groupID);

Параметры:

groupID

Идентификатор группы атрибутов, состав которой нам нужно получить. Если groupID = -1, то коллекция будет представлять список всех типов атрибутов, зарегистрированных в системе.

```
 IUserSession.GetAttributeTypeCollection(int groupID, bool filterRecs);
```

Параметры:

groupID

Идентификатор группы атрибутов, состав которой нам нужно получить.

filterRecs

Если значение параметра filterRecs равно *true*, то из списка исключаются типы атрибутов, которые текущий пользователь не может видеть.

3.5.2.2 Методы интерфейса

Create

Создает тип атрибута и возвращает его идентификатор.

Синтаксис:

```
int Create(AttributeTypeProperties attrProperties);
```

Параметры:

attrProperties

Структура, содержащая свойства создаваемого атрибута. Полю attrProperties.AttributeID можно присвоить 0, т.к. при создании атрибута значение этого поля не учитывается, поскольку сервер сам генерирует идентификатор для нового атрибута. Полю attrProperties.AttributeGuid можно присвоить значение Guid.Empty – в этом случае сервер сам сгенерирует глобальный идентификатор для создаваемого атрибута. Поле attrProperties.Name должно содержать непустое уникальное имя атрибута длиной до 250 символов, а поле attrProperties.Alias – или пустую строку, или уникальный псевдоним атрибута (также не более 250 символов).

Возвращаемое значение:

Локальный идентификатор созданного типа атрибута. В случае возникновения ошибок в процессе создания типа атрибута метод генерирует исключения.

GetAttributeTypeList

Метод возвращает массив обработчиков типов атрибутов по их идентификационным параметрам, хранящимся в idList.

Синтаксис:

```
IDBAttributeType[] GetAttributeTypeList(object[] idList, bool failIfNotFound);
```

Параметры:

idList

Список идентификаторов атрибутов. В качестве идентификаторов можно использовать полные наименования типов атрибутов, а также их глобальные и локальные идентификаторы. Метод идентификации система различает по типу идентифицирующего объекта (string, Guid или int).

failIfNotFound

Если значение параметра failIfNotFound равно *true*, то генерируется исключение в случае, когда запрашивается несуществующий тип атрибута, иначе несуществующие типы просто не попадут в результирующий массив.

Возвращаемое значение:

Массив интерфейсов, представляющих обработчики запрошенных типов атрибутов. Если какой-либо тип атрибута присутствует в idList более одного раза, то и в результирующем массиве его обработчик будет присутствовать соответствующее количество раз.

Select

Метод возвращает таблицу со списком типов атрибутов из текущей группы атрибутов.

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка **F_ATTRIBUTE_TYPE DESC, F_NAME DESC** отсортирует список атрибутов по типу хранимых данных и наименованию в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_ATTRIBUTES.

addInfo

Необязательный параметр, позволяющий управлять фильтрацией списка атрибутов по типу данных атрибута. Например, вызов `Select(String.Empty, FieldTypes.ftString)` вернет только строковые атрибуты.

Возвращаемое значение:

Таблица со списком атрибутов.

Таблица `IMS_ATTRIBUTES`. Список атрибутов.

Имя поля	Тип поля	Комментарий	IDBAttributeType
F_ATTRIBUTE_ID	INTEGER	Идентификатор атрибута.	AttributeID
F_NAME	NVARCHAR(255)	Наименование атрибута.	Name
F_SHORT_NAME	NVARCHAR(255)	Краткое наименование атрибута.	ShortName
F_ALIAS	NVARCHAR(255)	Псевдоним атрибута.	Alias
F_NOTE	NVARCHAR(450)	Комментарии.	Note
F_ATTRIBUTE_TYPE	INTEGER	Тип данных атрибута.	AttributeType
F_DEFAULT_VALUE	NVARCHAR(450)	Значение по умолчанию.	DefaultValue
F_MULTIPLE_VALUED	SMALLINT	Определяет, является ли данный атрибут списковым или многозначным.	MultipleValued
F_COMPUTED	SMALLINT	Способ вычисления значения атрибута.	Computed
F_SIZE_TYPE	INTEGER	Смысл поля зависит от типа данных атрибута (например, для строк содержит максимально допустимую длину строки).	SizeType
F_FORMULA	NVARCHAR(450)	Формула вычисления значения атрибута.	Formula
F_GUID	VARCHAR(40)	Глобальный идентификатор типа атрибута.	IDBGuid.GUID
F_AREA_ID	NVARCHAR(8)	Предметные области, к которым относится данный атрибут.	IDBSubjectArea. SubjectAreas
F_UNIQUE	SMALLINT	Контроль уникальности атрибута в пределах базы данных.	UniqueMode
F_LANGUAGE_ID	CHAR(1)	Идентификатор языкового варианта.	IDBLanguage. LanguageID
F_LEVEL_ID	INTEGER	Уровень продвижения, на котором может существовать данный атрибут.	LevelID
F_INVIEW	SMALLINT	Способ оптимизации записи/чтения атрибутов в общей таблице со списком объектов.	OptimizationMode
F_CONTENT	SMALLINT	Влияет ли данный атрибут на содержимое объекта (0 - нет, 1 - да).	IsContent
F_OPTIONS	INTEGER	Флаги с настройками атрибута.	Options
F_MASK	NVARCHAR(450)	Маска ввода значения атрибута.	Mask
F_MASTER_ID	INTEGER	Идентификатор мастер-атрибута. Если идентификатор больше 0, то это подчиненный атрибут (его значение зависит от значения мастер-атрибута).	MasterAttributeID
F_SOURCE_ID	INTEGER	Идентификатор атрибута, из которого подчиненный атрибут выбирает данные у объекта-источника.	SourceAttributeID

3.5.3 IDBAttributeType

Интерфейс для работы с типом атрибута.

3.5.3.1 Способы получения интерфейса

```
IUserSession.GetAttributeType(int attributeTypeID);
IUserSession.GetAttributeType(int attributeTypeID, bool failIfNotFound);
IUserSession.GetAttributeType(Guid attributeGuid);
IUserSession.GetAttributeType(Guid attributeGUID, bool failIfNotFound);
IUserSession.GetAttributeType(string attributeName);
IUserSession.GetAttributeType(string attributeName, bool failIfNotFound);
```

Параметры:

`attributeTypeID`

Локальный идентификатор атрибута.

`attributeGUID`

Глобальный идентификатор атрибута.

`attributeName`

Наименование атрибута. Нечувствительно к регистру букв.

`failIfNotFound`

Если параметр содержит `true`, то при запросе несуществующего типа атрибута методы возвращают `null`, иначе генерируют исключение `KernelException`. Методы без этого параметра также генерируют исключение `KernelException` в случае если запрашиваемый атрибут в системе не зарегистрирован.

```
IDBAttribute.AttributeType { get; }
```

Данное свойство экземпляра атрибута позволяет получить интерфейс соответствующего ему типа атрибута, причем свойства этого типа будут соответствовать настройкам типа атрибута применительно к объекту или связи, которым принадлежит данный экземпляр атрибута. Подробнее смотри описание данного свойства у интерфейса `IDBAttribute` (п. [4.3.2](#)).

3.5.3.2 Свойства интерфейса

Alias

Псевдоним атрибута. Может использоваться для альтернативной идентификации типов атрибутов. Может содержать уникальную строку длиной до 250 символов или пустое значение.

Синтаксис:

```
string Alias { get; set; }
```

AttributeID

Локальный идентификатор атрибута (только для чтения).

Синтаксис:

```
int AttributeID { get; }
```

AttributeType

Тип данных атрибута. Допустимые в системе типы данных описаны в Приложении 1. Там же приведена таблица допустимости преобразования одного типа данных в другой, а также присвоения значений атрибутов с различными типами данных.

Синтаксис:

```
FieldTypes AttributeType { get; set; }
```

ComputableAttribute

Свойство возвращает значение `true`, если данный атрибут может быть вычисляемым и участвовать в формулах вычисления и проверки значений других атрибутов.

Синтаксис:

```
bool ComputableAttribute { get; }
```

Computed

Способ вычисления атрибута.

Синтаксис:

```
ComputeValueModes Computed { get; set; }
```

Перечислитель `ComputeValueModes` может принимать одно из значений:

- `ComputeValueModes.NotComputableValue` – невычисляемый атрибут;
- `ComputeValueModes.StoredValue` – атрибут вычисляется в момент изменения значений атрибутов, участвующих в формуле вычисления значения данного атрибута. Поскольку вычисленное таким

образом значение хранится в базе данных, то этот способ вычисления не снижает скорости чтения значений атрибута, а также позволяет производить поиск информации по вычисленным значениям;

- `ComputeValueModes.JITValue` – атрибут вычисляется в момент чтения данных (такие атрибуты не могут быть использованы в выборках для поиска данных);
- `ComputeValueModes.IndexValue` – нормализованный индекс. Является вычисляемым атрибутом с хранением значений в базе данных, причем значение атрибута хранится в нормализованном виде (см. п. 3.1.3 в «Руководстве администратора»).

DefaultValue

Значение атрибута по умолчанию.

Синтаксис:

```
object DefaultValue{ get; set; }
```

Тип значения зависит от типа данных атрибута:

`FieldTypes`.

DefaultValueDescription

Возвращает текстовую расшифровку значения по умолчанию `DefaultValue`. Например, для типа `FieldTypes.fitObjectLink` свойство вернёт заголовок объекта, идентификатор которого записан в `DefaultValue`.

Синтаксис:

```
string DefaultValueDescription { get; }
```

EnabledOperators

Возвращает список операторов поиска, применимых к атрибуту данного типа (см. п. [5.1.7](#)). Набор допустимых операторов зависит от типа данных, способа вычисления атрибута, списка допустимых значений и пр.

Синтаксис:

```
RelationalOperators[] EnabledOperators { get; }
```

Formula

Формула вычисления значения атрибута. Правила записи формул приведены в Приложении 5 Руководства администратора. В формулах могут использоваться только атрибуты того же объекта (связи), к которому принадлежит вычисляемый атрибут.

Синтаксис:

```
string Formula { get; set; }
```

IsContent

Свойство позволяет указать, используется ли данный тип атрибута для хранения содержимого объекта. Изменение значения такого атрибута изменяет значение атрибута “Дата модификации содержимого объекта”, если у данного объекта такой атрибут имеется.

Синтаксис:

```
bool IsContent { get; set; }
```

IsGridable

Возвращает значение *true*, если атрибут может быть в качестве отображаемого атрибута в списках объектов или связей.

Синтаксис:

```
bool IsGridable { get; }
```

LevelID

Идентификатор уровня продвижения, при переходе на который объекту будет добавлен (или может быть добавлен) данный атрибут.

Синтаксис:

```
int LevelID { get; set; }
```

Mask

Маска ввода значения атрибута.

Синтаксис:

```
string Mask { get; set; }
```

MasterAttributeID

Идентификатор мастер-атрибута, для которого данный атрибут является подчиненным атрибутом. Содержит 0, если данный атрибут не является подчиненным атрибутом. Мастер-атрибутом может быть только атрибут типа **Ссылка на версию объекта** или **Ссылка на версию объекта без конкретизации** (FieldTypes.ftObjectLink).

Синтаксис:

```
int MasterAttributeID { get; set; }
```

MultipleValued

Свойство определяет, может ли атрибут принимать множественные значения, и ограничен ли он списком допустимых значений.

Синтаксис:

```
MultiValueModes MultipleValued { get; set; }
```

Перечислитель может принимать одно из следующих значений:

- MultiValueModes.SingleValue – атрибут может содержать одно значение;
- MultiValueModes.MultiValues – атрибут может содержать множество значений;
- MultiValueModes.SingleValueFromList – атрибут может содержать одно значение из списка разрешенных значений;
- MultiValueModes.MultiValuesFromList – атрибут может содержать множество значений из списка разрешенных значений.

Список разрешенных значений можно присвоить атрибуту с помощью метода SetPossibleValues (см. ниже). Значения SingleValueFromList и MultiValuesFromList можно присвоить только в том случае, если в базе данных нет объектов и связей, содержащих данный атрибут с неразрешенным значением. Пустое значение также всегда считается разрешенным, если это не противоречит флагу AttributeOptions.DisableNulls данного атрибута.

Внимание! Следует учитывать, что на атрибуты с множественным значением налагается ряд ограничений:

- в таблицах со списками объектов для таких атрибутов присутствует только первое значение – соответственно, сортировка выполняется только по этому значению;
- далеко не все условия поиска применимы к таким атрибутам;
- такие атрибуты не поддаются оптимизации на поиск информации, поскольку в оптимизационной таблице хранится только первое значение атрибута.

Name

Уникальное наименование атрибута. Содержит до 250 символов и не может принимать пустых значений. Нечувствительно к регистру букв.

Синтаксис:

```
string Name { get; set; }
```

Note

Комментарии к атрибуту (до 450 символов).

Синтаксис:

```
string Note { get; set; }
```

OptimizationMode

Задаёт способ оптимизации операций с атрибутом.

Синтаксис:

```
OptimizationModes OptimizationMode { get; set; }
```

Может принимать одно из следующих значений:

- OptimizationModes.Write – оптимизация на запись атрибута;
- OptimizationModes.Read – оптимизация на чтение атрибута;
- OptimizationModes.Seek – оптимизация на сортировку и поиск по атрибуту.

Внимание! Не рекомендуется без крайней необходимости использовать способы оптимизации Read и Seek для атрибута в глобальном списке типов атрибутов. Данные способы оптимизации обычно используют при применении атрибута к типу объекта или связи.

Options

Набор флагов, управляющих поведением атрибута.

Синтаксис:

```
AttributeOptions Options { get; set; }
```

Может содержать следующие флаги:

- `AttributeOptions.SaveInLog` – включает режим записи в системный журнал регистрации событий информации о всех модификациях значений данного атрибута;
- `AttributeOptions.SavePrivateHistory` – включает режим записи персональной истории изменения значений атрибута (пользователи смогут удалять из базы свою историю изменения значений атрибута);
- `AttributeOptions.SaveCommonHistory` – включает режим записи общей истории изменения значений атрибута (удаляемая пользователями персональная история сохраняется в базе для просмотра общей истории изменения значений атрибута);
- `AttributeOptions.DisableNulls` – позволяет запретить присвоение атрибуту пустых значений;
- `AttributeOptions.GetDescriptionEvent` – включает режим вызова службы `IDictionaryServerService` для расшифровки описания атрибута (при чтении значения `IDBAttribute.Description`);
- `AttributeOptions.Internal` – указывает клиентским редакторам на то, что данный атрибут используется для хранения системной информации и по умолчанию должен быть скрыт от пользователей;
- `AttributeOptions.ModifyInBase` – позволяет модифицировать значение атрибута без взятия на изменение объекта, которому принадлежит атрибут;
- `AttributeOptions.AddToGlobalIndex` – указывает системе на необходимость помещения значений данного атрибута в общий поисковый индекс;
- `AttributeOptions.EnableOwnerAccessCheck` – отключает у атрибута кэширование результатов проверки прав доступа для корректного вычисления прав, назначенных группе ВЛАДЕЛЕЦ_ОБЪЕКТА;
- `AttributeOptions.DontCopyPrototypeValue` – запрещает копирование значения атрибута в объект, создаваемый по прототипу данного объекта. Значение флага не распространяется на создание версий данного объекта;
- `AttributeOptions.DisableManualEdit` – запрещает пользователям редактировать значение атрибута вручную (в этом случае редактирование осуществляется только с помощью специализированных редакторов значений).

Примеры чтения и записи флагов в Options:

```
IDBAttributeType myAttributeType = session.GetAttributeType(myID);
// Так проверяем установлен ли флаг AttributeOptions.GetDescriptionEvent
if ((myAttributeType.Options & AttributeOptions.GetDescriptionEvent) ==
AttributeOptions.GetDescriptionEvent)
{
    ...
}
// Записываем флаги AttributeOptions.SaveInLog и AttributeOptions.DisableNulls
myAttributeType.Options = AttributeOptions.SaveInLog & AttributeOptions.DisableNulls;
```

PropertiesStructure

Позволяет прочитать или изменить свойства атрибута посредством структуры `AttributeTypeProperties`.

Синтаксис:

```
AttributeTypeProperties PropertiesStructure { get; set; }
```

При изменении свойств посредством данной структуры система проверяет значение поля `AttributeTypeProperties.AttributeID` – там должен содержаться правильный идентификатор данного атрибута, иначе будет сгенерировано исключение `KernelException`.

SizeType

Смысловое значение данного свойства зависит от типа данных атрибута:

- `ftString` – максимальная длина строки, которую можно записать в данный атрибут (не более `Intermech.Consts.MaxStringSize`);
- `ftMemo` – максимальный размер текста (не более `Intermech.Consts.MaxShortBlobSize` символов);
- `ftShortBlob` – максимальный размер двоичных данных (не более `Intermech.Consts.MaxShortBlobSize` байт);
- `ftObjectLink` – идентификатор типа объектов, которые можно присваивать данному атрибуту (содержит 0 если атрибуту можно присваивать ссылки на объекты любого типа);
- `ftMeasured` – идентификатор объекта типа **Физическая величина**, который содержит допустимые для данного атрибута единицы измерения (содержит 0 если данный атрибут может принимать значения, выраженные в любых единицах измерения);
- `ftExternalLink` – идентификатор объекта, описывающего источник данных из внешней БД.

Для остальных типов данных свойство должно содержать 0.

Синтаксис:

```
Int64 SizeType { get; set; }
```

SizeTypeDescription

Возвращает текстовую расшифровку значения свойства `SizeType` (например, наименование физической величины для атрибута типа `ftMeasured`).

Синтаксис:

```
string SizeTypeDescription { get; }
```

ShortName

Краткое имя атрибута (до 32 символов). Может содержать пустые и неуникальные значения.

Синтаксис:

```
string ShortName { get; set; }
```

SourceAttributeID

Идентификатор атрибута, из которого данный атрибут будет выбирать данные при присвоении значения мастер-атрибуту. Содержит 0 в случае, если данный атрибут не является подчиненным атрибутом.

Синтаксис:

```
int SourceAttributeID { get; set; }
```

UniqueMode

Метод контроля уникальности значений атрибута (имеет смысл только для атрибутов информационных объектов).

Синтаксис:

```
UniqueValueModes UniqueMode { get; set; }
```

Может принимать одно из следующих значений:

- `UniqueValueModes.NotUnique` – не контролировать уникальность;
- `UniqueValueModes.TypeOnly` – контролировать среди объектов данного типа;
- `UniqueValueModes.VerTypeOnly` – контролировать среди объектов данного типа и версий данного объекта;
- `UniqueValueModes.AllVerTypes` – контролировать среди всех объектов и их версий.

ValueFieldName

Имя поля базы данных, которое используется для хранения значения атрибута (например, для строковых атрибутов возвращает F_STRING_VALUE). Для комплексных атрибутов содержимого этого поля может быть недостаточно, т.к. там значение хранится в нескольких полях.

Синтаксис:

```
string ValueFieldName { get; }
```

ValidationRule

Условие правильности вводимых в атрибут значений (до 450 символов). Синтаксис условий описан в Приложении 1 Руководства администратора. Свойство может иметь значение только в случае, если данный интерфейс получен у атрибута применительно к типу объекта или связи – в глобальном списке атрибутов такого свойства атрибуты не имеют.

Синтаксис:

```
string ValidationRule { get; set; }
```

3.5.3.3 Методы интерфейса

Delete

Удаляет данный тип атрибута из базы данных. Для удаления типа атрибута из общего списка атрибутов нужно, чтобы данный тип атрибут не был назначен типам объектов и связей, а также в базе не было ни одного объекта или связи с экземпляром данного атрибута. Также не должно быть ссылок на данный атрибут из формул и подчиненных атрибутов.

Синтаксис:

```
Delete(Int64 DeleteMode)
```

Параметры:

DeleteMode

При удалении типа атрибута из общего списка атрибутов параметр не используется. При удалении атрибута у типа объектов или связей может содержать флаг Intermech.Consts.DeleteInstances, который указывает серверу на необходимость удалить соответствующий экземпляр атрибута у всех объектов или связей данного типа. Удаление экземпляров атрибутов производится без проверок прав доступа, взятия объектов на изменение и пр., поэтому не следует использовать данный флаг без крайней необходимости. По этой причине данный флаг не может быть применён к атрибутам, влияющим на дату модификации содержимого объектов (IDBAttributeType.IsContent = true).

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

GetGroupsList

Возвращает массив идентификаторов групп, в которые входит данный атрибут (за исключением системной группы **Все атрибуты**).

Синтаксис:

```
int[] GetGroupsList();
```

GetPossibleValues

Возвращает таблицу допустимых значений атрибута.

Синтаксис:

```
DataTable GetPossibleValues();
```

Возвращаемое значение:

Таблица из следующих полей:

1. Порядковый номер значения, начиная с 0 (поле F_INLIST_ID).
2. Допустимое значение. Имя поля и тип данных зависит от типа данных атрибута (например, для целочисленного атрибута содержит поле F_INTEGER_VALUE, значение которого приводится к Int64).
3. Необязательная строковая расшифровка допустимого значения (длиной до 450 символов).
Используется для представления значений в удобном для восприятия пользователей виде.

Таблица не содержит записей в случае, если данный атрибут не имеет допустимых значений.

GetPossibleValuesArray

Возвращает массив допустимых значений атрибута в порядке их следования в списке допустимых значений. Тип значений зависит от типа данных атрибута. Если атрибут не содержит допустимых значений, то функция возвращает массив нулевой длины.

Синтаксис:

```
object[] GetPossibleValuesArray();
```

IsCompatibleType

Функция проверяет можно ли изменить тип данных атрибута на тип newType.

Синтаксис:

```
bool IsCompatibleType(FieldTypes newType);
```

Параметры:

newType

Проверяемый тип данных.

Возвращаемое значение:

Возвращает *true*, если тип newType можно преобразовать к данному типу.

SetPossibleValues

Присваивает атрибуту новый список допустимых значений.

Синтаксис:

```
void SetPossibleValues ( DataTable valuesTable );
```

Параметры:

valuesTable

Таблица присваиваемых атрибуту допустимых значений. Структура таблицы должна быть аналогична таблице, возвращаемой методом GetPossibleValues для данного типа атрибута.

ValidateAssign

Функция проверяет возможность присвоения значений данному типу атрибута из типа атрибута source. Если это в принципе невозможно, то выдается исключение *KernelExceptionID*.

Синтаксис:

```
void ValidateAssign(IDBAttributeType source);
```

Параметры:

source

Тип атрибута, который проверяется на допустимость использования в качестве источника значений.

3.6 Типы объектов

Тип объекта – это главная характеристика любого информационного объекта, которая определяет:

- допустимый набор атрибутов, которые могут присваиваться объекту автоматически или вручную;
- допустимые связи с другими объектами;
- этапы жизненного цикла, на которые можно переводить данный объект (и, соответственно, права доступа к объекту);
- набор команд локального меню, которые могут быть применены к данному объекту;
- способы обработки объекта различными модулями системы и т.д.

Пространство имен: *Intermech.Interfaces*.

Интерфейс для работы со списком типов объектов: *IDBObjectTypeCollection*.

Интерфейс для работы с типом объекта: *IDBObjectType*.

3.6.1 ObjectTypeProperties

Структура, позволяющая описать все свойства типа объектов. Поля структуры соответствуют свойствам интерфейса *IDBObjectType* (см. п. 3.6.3). Кроме основного конструктора, в который нужно передать значения всех полей структуры, имеется также дополнительный конструктор, позволяющий инициализировать структуру строкой таблицы с описателем типа объектов *IMS_OBJECT_TYPES*.

3.6.2 IDBObjectTypeCollection

Интерфейс позволяет создавать новые типы объектов, получать список уже существующих типов объектов, проверять существование типов объектов с указанными идентификаторами и пр. Реализация осуществляется объектом *IDBObjectTypeCollection*.

3.6.2.1 Способы получения интерфейса

IUserSession.GetObjectTypeCollection (int parentTypeID);

Параметры:

parentTypeID

Идентификатор родительского типа объектов, с дочерними типами которого будет работать полученный интерфейс. Если параметр parentTypeID = -1, то метод возвращает интерфейс, работающий с корневыми типами объектов, если parentTypeID = -2, то интерфейс работает со всеми типами объектов, зарегистрированных в системе.

`IUserSession.GetObjectTypeCollection(int parentTypeID, bool filterRecs);`

Параметры:

parentTypeID

Идентификатор родительского типа объектов.

filterRecs

Если значение параметра filterRecs равно **true**, то при вызове метода Select из списка исключаются типы объектов, которые данный пользователь не может видеть.

3.6.2.2 Методы интерфейса

Create

Метод создает новый тип объектов и возвращает его идентификатор. Если коллекция типов была получена с неотрицательным идентификатором родительского типа, то созданный тип объекта будет автоматически включен в состав указанного родительского типа объектов.

Синтаксис:

`int Create(ObjectTypeProperties typeProperties);`

Параметры:

typeProperties

Структура, содержащая свойства типа объектов (см. п. 3.6.1). В поле ObjectType укажите 0. В поле ObjectTypeGuid можно записать Guid.Empty – тогда сервер сам сгенерирует новый глобальный идентификатор для создаваемого типа объектов.

GetTypesHierarchy

Возвращает таблицу с полной иерархией наследования всех типов объектов. Данная версия ядра системы поддерживает только единичное наследование типов объектов, но в дальнейших возможна поддержка множественного наследования типов.

Синтаксис:

`DataTable GetTypesHierarchy();`

Таблица IMS_OBJECT_TYPES_TREE. Иерархия типов объектов.

Имя поля	Тип поля	Комментарий	IDBObjectType
F_PARENT_ID	INTEGER	Идентификатор родительского типа объекта.	ParentTypeID
F_OBJECT_TYPE	INTEGER	Идентификатор дочернего типа объекта.	ObjectType

GetUsedByAttribute

Возвращает таблицу типов объектов, для которых задано использование атрибута с идентификатором attributeID.

Синтаксис:

`DataTable GetUsedByAttribute(int attributeID);`

Параметры:

attributeID

Идентификатор атрибута.

Формат таблицы аналогичен формату таблицы IMS_OBJECT_TYPES (см. ниже описание метода Select).

Select

Метод возвращает таблицу со списком типов объектов, .

Синтаксис:

`DataTable Select (string orderBy, params object[] addInfo);`

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка **F_OBJ_TYPE_NAME DESC** отсортирует список типов объектов по их наименованию в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_OBJECT_TYPES.

addInfo

Если передать параметр true, то в возвращаемой таблице часть полей будет заменена их текстовой расшифровкой (например, вместо идентификаторов атрибутов-заголовков в таблице будут содержаться их наименования).

Возвращаемое значение:

Таблица со списком типов объектов.

Таблица IMS_OBJECT_TYPES. Список типов объектов.

Имя поля	Тип поля	Комментарий	IDBObjectType
F_OBJECT_TYPE	INTEGER	Идентификатор типа объекта.	ObjectType
F_OBJECT_TYPE_NAME	NVARCHAR(255)	Наименование типа объектов (например, “Детали”).	ObjectTypeName
F_OBJ_NAME	NVARCHAR(255)	Наименование объекта данного типа (например, “Деталь”).	ObjectInstanceName
F_ICON	IMAGE	Иконка для отображения типа объектов.	Icon
F_VERSIONABLE	SMALLINT	Управление версионностью типов объектов.	Versionable
F_NOTE	NVARCHAR(450)	Комментарии.	Note
F_DEFAULT_RELATION	INTEGER	Идентификатор типа связи, показываемой в клиенте по умолчанию.	DefaultRelation
F_GUID	VARCHAR(40)	Глобальный уникальный идентификатор типа объектов. По нему будет работать портфель при загрузке и синхронизации базы.	IDBGuid.GUID
F_AREA_ID	NVARCHAR(8)	Предметные области, к которым относится данный тип объектов.	IDBSubjectArea. SubjectAreas
F_CAPTION_ATTRIBUTE	INTEGER	Идентификатор атрибута, который используется для отображения объектов данного типа в списках и деревьях.	CaptionAttribute
F_ANY_ATTRIBUTES	SMALLINT	Контроль набора атрибутов: 0 - допускается добавлять к объектам данного типа только разрешенные атрибуты. 1 - допускается добавлять любые атрибуты.	AnyAttributes
F_PUBLIC_LC	SMALLINT	Наследование схемы жизненного цикла (0 – собственная схема, 2 – унаследованная от родительского типа).	PublicLC
F_SHORT_NAME	NVARCHAR(32)	Краткое наименование типа объектов.	ObjectTypeShortName
F_DEL_TIME	INTEGER	Количество дней, в течение которых нельзя чистить удаленные объекты данного типа.	LifeTimeReserve

SelectRecursive

Возвращает рекурсивный список всех дочерних типов объектов до последнего уровня вложенности.

DataTable SelectRecursive(string orderBy, params object[] addInfo);

Формат таблицы аналогичен формату таблицы IMS_OBJECT_TYPES (см. описание метода IDBObjectTypeCollection.Select).

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Допустимые имена полей см. в таблице IMS_OBJECT_TYPES.

addInfo

Если передать параметр `true`, то в возвращаемой таблице часть полей будет заменена их текстовой расшифровкой (например, вместо идентификаторов атрибутов-заголовков в таблице будут содержаться их наименования).

Возвращаемое значение:

Таблица со списком всех дочерних типов объектов до последнего уровня вложенности.

3.6.3 IDBObjectType

Интерфейс предназначен для работы с типом объектов. Позволяет читать и модифицировать его свойства, получать список атрибутов, назначенных типу объектов, добавлять и удалять подтипы, а также удалять сам тип объектов. Данный интерфейс унаследован от интерфейса `IDBAttributebleType`, который содержит общие методы и свойства для работы с атрибутами, назначенными типам объектов и связей.

3.6.3.1 Способы получения интерфейса

```
IUserSession.GetObjectType(int objectTypeID);
IUserSession.GetObjectType(int objectTypeID, bool failIfNotFound);
IUserSession.GetObjectType(string objectTypeName);
IUserSession.GetObjectType(string objectTypeName, bool failIfNotFound);
IUserSession.GetObjectType(Guid objectTypeGuid);
IUserSession.GetObjectType(Guid objectTypeGuid, bool failIfNotFound);
```

Параметры:

`objectTypeID`

Локальный идентификатор типа объектов.

`objectTypeName`

Уникальное наименование типа объектов.

`objectTypeGuid`

Глобальный идентификатор типа объектов.

`failIfNotFound`

Если данный параметр содержит **true**, то функции будут генерировать исключение `KernelExceptionID` при отсутствии в базе типа объектов с указанным локальным идентификатором (номер ошибки 226), наименованием (номер ошибки 97) или глобальным идентификатором (номер ошибки 99). Функции без данного параметра (или со значением параметра **false**) в случае отсутствия в базе запрошенного типа объектов возвращают `null`.

3.6.3.2 Свойства интерфейса

AnyAttributes

Свойство позволяет присваивать объектам данного типа любые атрибуты. При этом если у типа объектов данный тип атрибута не зарегистрирован, то характеристики атрибута (уникальность, формула вычисления значений, регистрация истории значений и пр.) берутся из глобального списка типов атрибутов, зарегистрированных в системе. Выключение данной опции возможно только в том случае, если в базе данных у всех объектов данного типа имеются только разрешенные атрибуты.

Синтаксис:

```
bool AnyAttributes { get; set; }
```

Attributes

Коллекция типов атрибутов, разрешенных для данного типа объектов. Все атрибуты из данной коллекции имеют собственный набор характеристик применительно к данному типу объектов. Данная коллекция может быть приведена к интерфейсу `IDBAttribute4ObjectTypeCollection` и может использоваться для добавления данному типу объектов разрешенных атрибутов, а также для поиска уже добавленных типов атрибутов.

Синтаксис:

```
IDBAttribute4TypeCollection Attributes { get; }
```

CaptionAttribute

Идентификатор атрибута, который используется в качестве заголовка для объектов данного типа. Заголовок используется для отображения объектов в интерфейсе клиентской части системы. Если свойство содержит 0, то заголовки объектов заполняются независимо от остальных атрибутов. В качестве заголовка не могут выступать обязательные атрибуты объектов (с идентификатором меньше 0). В качестве заголовка могут выступать только атрибуты, разрешенные для данного типа объектов (даже если тип допускает наличие у объектов любых атрибутов) и имеющие строковый или ссылочный тип данных.

Синтаксис:

```
int CaptionAttribute { get; set; }
```

DefaultRelation

Идентификатор типа связи, который используется в универсальном клиенте для раскрутки состава объектов данного типа. В дополнение к данной настройке администратор может назначать типы связей для ролей (например, для роли “Технолог” в составе сборочных единиц кроме связей между изделиями могут просматриваться также и технологические связи). Для получения полного списка видимых на клиенте типов связей используется механизм отображения и сортировки составов (см. п. [8.12.8](#)).

Синтаксис:

```
int DefaultRelation { get; set; }
```

Icon

Иконка для отображения типа объектов.

Синтаксис:

```
byte[] Icon { get; set; }
```

LifetimeReserve

Время жизни удаленных объектов - количество дней, в течение которых нельзя физически уничтожать объекты данного типа, переведенные на уровень продвижения “Удалено”. Если свойство содержит 0, то объекты при удалении в системе будут сразу удаляться и из базы данных.

Синтаксис:

```
int LifetimeReserve { get; set; }
```

Note

Комментарии к типу объекта (до 450 символов).

Синтаксис:

```
string Note { get; set; }
```

ObjectInstanceName

Наименование объекта данного типа (например, **Деталь**). Должно обязательно содержать значение (до 255 символов).

Синтаксис:

```
string ObjectInstanceName { get; set; }
```

ObjectType

Локальный идентификатор типа объекта (только для чтения). Назначается системой автоматически в момент создания типа объектов. В разных базах данных тип объекта может иметь разные локальные идентификаторы.

Синтаксис:

```
int ObjectType { get; }
```

ObjectTypeName

Наименование типа объектов (например, “Детали”). Должно обязательно содержать уникальное значение (до 255 символов).

Синтаксис:

```
string ObjectTypeName { get; set; }
```

ObjectTypeShortName

Краткое наименование типа объектов. Может содержать пустое значение. Непустое значение должно быть уникальным и может содержать до 32 символов.

Синтаксис:

```
string ObjectTypeShortName { get; set; }
```

ParentTypeID

Идентификатор типа объектов, от которого унаследован данный тип. Для корневых типов содержит -1.

Синтаксис:

```
int ParentTypeID { get; set; }
```

PropertiesStructure

Свойство, позволяющее прочитать или записать сразу все параметры типа объектов.

Синтаксис:

```
ObjectTypeProperties PropertiesStructure { get; set; }
```

PublicLC

Свойство в данной версии не используется.

SchemaID

Идентификатор схемы жизненного цикла для объектов данного типа.

Синтаксис:

```
int SchemaID { get; set; }
```

Versionable

Свойство управление версионностью объектов данного типа.

Синтаксис:

```
ObjectVersionModes Versionable { get; set; }
```

Может принимать следующие значения:

- ObjectVersionModes.Abstract – абстрактный тип объектов (в базе данных нельзя создавать объекты данного типа);
- ObjectVersionModes.SingleVersion – неверсионный тип объектов (нельзя создавать более одной версии объекта данного типа);
- ObjectVersionModes.MultiVersion – версионный тип объектов (можно создавать сколько угодно версий объекта данного типа).

Примечание: при изменении версионности на абстрактный или неверсионный тип система проверяет наличие в базе информационных объектов данного типа и отменяет операцию при наличии в базе версий объектов, несоответствующих данному типу версионности.

VisibleAttributes

Список допустимых атрибутов для данного типа объектов, отфильтрованный в соответствии с правами доступа и настройками профиля пользователя.

Синтаксис:

```
IDBAttribute4TypeCollection VisibleAttributes { get; }
```

3.6.3.3 Методы интерфейса

Delete

Метод удаляет данный тип объектов. Для выполнения операции в базе данных не должно быть экземпляров объектов данного типа, а у типа не должно быть подтипов.

Синтаксис:

```
int Delete(Int64 DeleteMode);
```

Параметры:

DeleteMode

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

GetPossibleChildren

Метод возвращает хэш-таблицу следующего содержания:

идентификатор типа объекта = идентификатор типа связи, где в качестве ключей приведены идентификаторы всех неабстрактных типов объектов, которые можно включать в состав объектов данного типа, а в качестве значений – соответствующий им типы связей.

Синтаксис:

```
Hashtable GetPossibleChildren();
```

IncludeObjectType

Метод перемещает в список дочерних типов объектов типы с номерами objectTypes. Операция выполняется только в том случае, если в базе данных еще нет объектов типа objectTypes (или объектов унаследованных типов).

Синтаксис:

```
int IncludeObjectType( int[] objectTypes );
```

Параметры:

objectTypes

Массив локальных идентификаторов типов объектов, которые нужно переместить в список дочерних типов для данного типа объектов.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок перемещения метод генерирует исключения.

3.6.4 Attribute4ObjectTypeProperties

Структура, описывающая свойства атрибута в контексте определенного типа объектов. Используется при добавлении атрибутов типам объектов, а также при чтении и модификации свойств атрибута применительно к типу объектов. Назначение свойств аналогично свойствам интерфейсов IDBAttributeType (см. п. [3.5.3](#)) и IDBAttribute4ObjectType (см. п. [3.6.6](#)).

3.6.5 IDBAttribute4ObjectTypeCollection

Интерфейс позволяет создавать новые атрибуты в контексте определенного типа объектов, получать список уже существующих атрибутов, а также проверять принадлежность атрибутов заданному типу объектов.

3.6.5.1 Способы получения интерфейса

Коллекция атрибутов применительно к типу объекта получается у интерфейса IDBObjectType с помощью свойств `Attributes` и `VisibleAttributes`, например:

```
IDBAttribute4ObjectTypeCollection attrs;  
// Получаем полную коллекцию атрибутов для типа объектов номер typeID  
attrs = session.GetObjectType(typeID).Attributes as IDBAttribute4ObjectTypeCollection;  
// Получаем фильтрованную коллекцию атрибутов (видимых для пользователя сессии session)  
attrs = session.GetObjectType(typeID).VisibleAttributes as IDBAttribute4ObjectTypeCollection;
```

3.6.5.2 Методы

Create

Добавляет атрибут к списку атрибутов данного типа объектов.

Синтаксис:

```
public IDBAttributeType4Object Create(Attribute4ObjectTypeProperties attrProperties);
```

Параметры:

`attrProperties`

Структура, описывающая свойства атрибута применительно к данному типу объектов.

Возвращаемое значение:

Интерфейс атрибута применительно к типу объектов (см. п. [3.6.6](#)).

GetAttributeByID

Метод ищет атрибут `attributeID` в списке атрибутов для данного типа объектов.

Синтаксис:

```
public IDBAttributeType4 GetAttributeByID(int attributeID, bool throwNotFoundException);
```

Параметры:

`attributeID`

Локальный идентификатор искомого атрибута.

`throwNotFoundException`

Если содержит **true**, то метод генерирует исключение при отсутствии искомого атрибута у данного типа объектов. Иначе при отсутствии атрибута будет возвращать `null`. Метод `GetAttributeID` без данного параметра также возвращает `null` в случае, если искомым атрибут не найден.

Возвращаемое значение:

Интерфейс, описывающий свойства атрибута применительно к типу объекта или связи. Может быть приведен к интерфейсу `IDBAttributeType4Object` (см. п. [3.6.6](#)).

GetDefaultProperties

Метод ищет структуру со свойствами по умолчанию для атрибута `attributeID` для данного типа объектов `Attribute4ObjectTypeProperties GetDefaultProperties(int attributeID);`

Параметры:

`attributeID`

Локальный идентификатор атрибута.

Возвращаемое значение:

Структура, описывающая свойства атрибута.

Select

Метод возвращает таблицу атрибутов для данного типа объектов. В таблице представлены все свойства атрибутов, которые могут быть изменены применительно к данному типу объектов (естественно, за исключением самих идентификаторов типов).

Синтаксис:

```
public DataTable Select (string orderBy, params object [] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка **F_ATTRIBUTE_ID DESC** отсортирует таблицу по идентификаторам атрибутов в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_ATTR4OBJ_TYPES.

addInfo

Если параметр содержит значение Intermech.Consts.GetAllFields, то результирующая таблица будет дополнена недостающими полями из таблицы атрибутов IMS_ATTRIBUTES (F_NAME, F_SHORT_NAME, F_NOTE, F_ATTRIBUTE_TYPE и пр.).

Возвращаемое значение:

Таблица со списком атрибутов для заданного типа объектов.

Таблица IMS_ATTR4OBJ_TYPES. Набор параметров для типа объектов.

Имя поля	Тип поля	Комментарий	IDBAttributeType4Object
F_ATTRIBUTE_ID	INTEGER	Идентификатор атрибута.	AttributeID
F_OBJECT_TYPE	INTEGER	Идентификатор типа объектов.	ObjectType
F_PUBLIC	SMALLINT	0 – атрибут не передаётся подтипам данного типа объектов; 1 - атрибут передаётся подтипам; 2 – атрибут унаследован от родительского типа.	InheritMode
F_REQUIRED	SMALLINT	Контроль обязательности атрибута для данного типа объектов.	Required
F_VALIDATION_RULE	NVARCHAR(450)	Условие контроля правильности значений, попадающий в этот атрибут в контексте данного типа объектов.	ValidationRule
F_COMPUTED	SMALLINT	Способ вычисления значения атрибута.	Computed
F_FORMULA	NVARCHAR(450)	Формула вычисления значения атрибута.	Formula
F_UNIQUE	SMALLINT	Контроль уникальности поля в пределах базы данных.	UniqueMode
F_LEVEL_ID	INTEGER	Уровень продвижения, на котором может существовать данный атрибут.	LevelID
F_DEFAULT_VALUE	NVARCHAR(450)	Значение атрибута по умолчанию.	DefaultValue
F_INVIEW	SMALLINT	Способ оптимизации записи/чтения атрибута у объектов данного типа.	OptimizationMode
F_CONTENT	SMALLINT	Влияет ли атрибут на содержимое объектов данного типа.	IsContent
F_OPTIONS	INTEGER	Флаги с настройками.	Options
F_MASK	NVARCHAR(450)	Маска ввода значения атрибута.	Mask
F_MASTER_ID	INTEGER	Идентификатор мастер-атрибута. Если идентификатор больше 0, то это подчиненный атрибут (его значение зависит от значения мастер-атрибута).	MasterAttributeID

F_SOURCE_ID	INTEGER	Идентификатор атрибута, из которого подчиненный атрибут выбирает данные у объекта-источника.	SourceAttributeID

3.6.6 IDBAttributeType4Object

Интерфейс, описывающий свойства атрибута в контексте определенного типа объектов. Наследует все методы и свойства интерфейса IDBAttributeType (см. п. [3.5.3 IDBAttributeType](#)). Часть свойств типа атрибута может быть изменена применительно к данному типу объектов (например, формула вычисления значения, способы оптимизации, контроль пустых значений и пр.). Свойства, которые не могут быть изменены (например, наименование и тип данных атрибута), выдают при попытке присвоения нового значения исключение OperationNotApplicableException.

Примечание: формула вычисления значения, условие контроля правильности значения, а также мастер-атрибут могут ссылаться только на атрибуты, уже присвоенные данному типу объекта!

Далее приведено описание свойств, специфичных для данного интерфейса.

3.6.6.1 Свойства интерфейса

Attribute4ObjectPropertiesStructure

Свойство позволяет прочитать или записать сразу все параметры атрибута в контексте данного типа объектов. При записи система проверяет правильность заполнения полей AttributeID и ObjectType – они должны содержать соответствующие идентификаторы атрибута и типа объектов.

Синтаксис:

```
public Attribute4ObjectTypeProperties Attribute4ObjectPropertiesStructure{ get; set; }
```

InheritMode

Правило передачи по наследству атрибута от родительских типов дочерним типам объектов.

Синтаксис:

```
public InheritModes InheritMode{ get; set; }
```

Свойства наследования атрибута могут быть следующими:

- InheritModes.Private – атрибут принадлежит только данному типу объекта;
- InheritModes
- .Public – атрибут передается по наследству дочерним типам объектов до их последнего уровня вложенности. Наследование прерывается в случае, если у дочернего типа уже есть аналогичный атрибут;
- InheritModes.Inherited – атрибут унаследован от родительского типа объектов.

Required

Свойство определяет обязательность атрибута для данного типа объектов, а также способ добавления атрибута к объектам данного типа.

Синтаксис:

```
public RequiredModes Required{ get; set; }
```

Может принимать следующие значения:

- RequiredModes.Manual – необязательный атрибут, который может быть добавлен вручную;
- RequiredModes.Auto – необязательный атрибут, добавляемый автоматически;
- RequiredModes.AutoRequired – обязательный атрибут, добавляемый автоматически.

Автоматическое добавление атрибутов происходит в момент создания заготовки объекта. Если у атрибута указан уровень, то атрибут добавляется к объекту в момент его перевода на шаг с данным уровнем продвижения. При добавлении к типу атрибута со значением свойства RequiredModes.Auto или RequiredModes.AutoRequired система добавляет данный атрибут ко всем существующим в базе объектам данного типа.

ValidationRule

Условие контроля правильности вводимых в атрибут значений.

Синтаксис:

```
public string ValidationRule{ get; set; }
```

В формуле допускаются ссылки на значения атрибутов данного объекта, заключенные в квадратные скобки. Значение самого атрибута представляется ключевым словом Value. Строковые константы записываются в одинарные кавычки. Условия могут быть объединены логическими операторами OR (или) и AND (и). Допускается использовать операторы сравнения <, >, <=, >=, <>, =, IN, LIKE. Также в формулах можно использовать функции LEN, ISNULL, IIF, TRIM, SUBSTRING .

Пример:

```
Value >= [Дата регистрации] AND Value <= [Дата аннулирования]
```

3.7 Схемы жизненных циклов объектов

Схемой жизненного цикла называют графическое представление последовательности допустимых шагов жизненного цикла объектов.

Пространство имен: Intermech.Interfaces.LifeCycles

3.7.1 IDBLCSchemaCollection

Интерфейс для работы со списком схем жизненных циклов, зарегистрированных в системе.

3.7.1.1 Методы интерфейса

Create

Создает схему жизненного цикла и возвращает ее локальный идентификатор.

Синтаксис:

```
int Create(DBLCSchemaProperties properties);
```

Параметры:

properties

Структура, описывающая свойства создаваемой схемы жизненного цикла.

GetDefaultSchemaID

Возвращает идентификатор схемы жизненного цикла, которая назначена как схема по умолчанию.

Синтаксис:

```
int GetDefaultSchemaID();
```

3.7.2 IDBLCSchema

Интерфейс для работы со схемой жизненного цикла объектов.

3.7.2.1 Свойства интерфейса

GUID

Глобальный идентификатор схемы ЖЦ.

Синтаксис:

```
Guid GUID { get; set; }
```

Name

Уникальное наименование схемы ЖЦ. Может содержать до 250 символов.

Синтаксис:

```
string Name { get; set; }
```

Note

Примечания к схеме ЖЦ. Может содержать до 450 символов.

Синтаксис:

```
string Note { get; set; }
```

IsDefaultSchema

Свойство позволяет определить является ли данная схема схемой ЖЦ по умолчанию. Схема по умолчанию присваивается вновь создаваемым типам объектов, которые размещаются на верхнем уровне иерархии типов.

Синтаксис:

```
bool IsDefaultSchema { get; set; }
```

Options

Опции схемы ЖЦ. Свойство зарезервировано для будущего использования.

Синтаксис:

```
LCSchemaOptions Options { get; set; }
```

SchemaID

Локальный идентификатор схемы ЖЦ.

Синтаксис:

```
int SchemaID { get; }
```

SchemaProperties

Структура со свойствами схемы ЖЦ. Позволяет одним вызовом прочитать или записать все свойства схемы.

Синтаксис:

```
DBLCSchemaProperties SchemaProperties { get; set; }
```

3.7.2.2 Методы интерфейса

Delete

Удаляет схему жизненного цикла объектов. Удалить можно схему, на шагах которой в данный момент нет ни одной версии объектов. Также в базе не должно быть типов объектов, которым назначена данная схема ЖЦ.

Синтаксис:

```
int Delete(Int64 deleteMode);
```

Параметры:

deleteMode

Зарезервировано для будущего использования.

GetStepsCollection

Возвращает идентификатор первого шага схемы.

Синтаксис:

```
IDBLifecycleStepCollection GetStepsCollection();
```

Возвращаемое значение:

Возвращает коллекцию этапов (шагов) жизненного цикла, из которых состоит данная схема.

3.8 Этапы жизненного цикла объектов

Этапом (шагом) жизненного цикла называется свойство объекта, определяющее текущее состояние объекта в процессе его жизненного цикла. От текущего шага жизненного цикла зависят права доступа к объекту и возможные способы его изменения, уровень продвижения объекта, способы обработки объекта различными модулями системы и пр.

Пространство имен: Intermech.Interfaces

3.8.1 IDBLifecycleStepCollection

Интерфейс коллекции этапов (шагов) жизненного цикла объектов, принадлежащих определенной схеме ЖЦ.

3.8.1.1 Свойства интерфейса

ObjectTypeID

Идентификатор типа объектов, которому принадлежит данная схема.

Синтаксис:

```
int ObjectTypeID { get; }
```

3.8.1.2 Методы интерфейса

Create

Создает этап жизненного цикла.

Синтаксис:

```
IDBLifecycleStep Create(DBLifecycleStepProperties lcProps);
```

Параметры:

lcProps

Структура, описывающая свойства шага жизненного цикла.

GetFirstStep

Возвращает идентификатор первого шага схемы.

Синтаксис:

```
int GetFirstStep();
```

Возвращаемое значение:

Идентификатор первого шага схемы.

GetSchema

Возвращает DATASET с таблицами IMS_LC_STEPS и IMS_LC_LINKS, в которых описаны шаги и связи схемы жизненного цикла для данного типа объектов (номер типа см. в ObjectTypeID).

Синтаксис:

```
ImDataSet GetSchema();
```

Возвращаемое значение:

Таблицы IMS_LC_STEPS и IMS_LC_LINKS.

Таблица IMS_LC_LINKS. Связи схемы жизненного цикла для типа объектов.

Имя поля	Тип поля	Комментарий	IDBLifecycleStep
F_FROM_STEP	INTEGER	Идентификатор этапа жизненного цикла.	LCStep
F_TO_STEP	INTEGER	Идентификатор этапа жизненного цикла.	LCStep
F_ROUTE_ID	INTEGER	Идентификатор маршрута (OBJECT_ID), который нужно выполнить для перемещения по данной связи.	RouteID
F_NOTE	NVARCHAR(450)	Комментарии к связи.	Note
F_PARAMS	INTEGER	Свойства связи (зарезервировано).	Properties

Таблица IMS_LC_STEPS. Этапы схемы жизненного цикла для типа объектов.

Имя поля	Тип поля	Комментарий	IDBLifecycleStep
F_LC_STEP	INTEGER	Идентификатор этапа жизненного цикла.	LCStep
F_LEVEL_ID	INTEGER	Идентификатор уровня продвижения.	LevelID
F_LC_NAME	NVARCHAR(255)	Наименование этапа жизненного цикла.	LCName
F_NOTE	NVARCHAR(450)	Комментарии к связи.	Note
F_OBJECT_TYPE	INTEGER	Идентификатор типа объекта, к которому относится данный этап жизненного цикла.	ObjectTypeID
F_ACCESS_TYPE	SMALLINT	Тип доступа к объектам на данном этапе жизненного цикла.	AccessType
F_DRAW_DATA	LARGE BINARY	Информация для отображения данного этапа жизненного цикла на схеме.	DrawData
F_DELETED	SMALLINT	Если значение поля не равно 0, то этап удален.	-
F_MODIFY_MODE	SMALLINT	Способ модификации объектов.	ObjectModifyMode
F_GUID	VARCHAR(40)	Глобальный идентификатор этапа жизненного цикла.	StepGuid
F_FIRST	BOOLEAN	Значение поля определяет, является ли данный этап первым в схеме жизненного цикла.	IsFirstStep

Select

Метод возвращает таблицу со списком этапов жизненного цикла объектов.

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка "F_LC_NAME DESC" отсортирует список типов связей по их наименованию в убывающем

порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_RELATION_TYPES.

`addInfo`

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица со списком типов связей IMS_LC_LINKS. Ее описание приведено в методе `GetSchema`.

`SetObjectsLCStep`

Изменяет шаг жизненного цикла у заданных объектов.

Синтаксис:

```
void SetObjectsLCStep(long[] objectIDs, int lcStep);
```

Параметры:

`objectIDs`

Массив идентификаторов объектов.

`lcStep`

Идентификатор шага жизненного цикла.

3.8.2 IDBLifecycleStep

Данный интерфейс используется для работы с этапом жизненного цикла данного типа объектов.

Реализация осуществляется объектом `DBLifecycleStep`.

3.8.2.1 Способы получения интерфейса

`IUserSession.GetLifecycleStep (int aLCStepID);`

Параметры:

`aLCStepID`

Идентификатор этапа жизненного цикла.

`IUserSession.GetLifecycleStep (Guid anLCGuid);`

Параметры:

`anLCGuid`

Глобальный идентификатор этапа жизненного цикла.

`IUserSession.GetLifecycleStep (Guid anLCGuid, bool throwException);`

Параметры:

`anLCGuid`

Глобальный идентификатор этапа жизненного цикла.

`throwException`

Если значение параметра равно **true**, то при ошибке генерируется исключение.

3.8.2.2 Свойства интерфейса

`AccessType`

Тип доступа к объектам на данном этапе жизненного цикла:

0 – контроль прав доступа не производится,

1 – контроль только по правам ЖЦ (без возможности индивидуального назначения прав),

2 – контроль по ЖЦ и объекту (без возможности передачи прав доступа по наследству),

3 – то же, но с возможностью публиковать права подтипам.

Синтаксис:

```
LCAccessTypes AccessType{ get; set; }
```

Контроль прав доступа к объектам:

- `LCAccessTypes.NoCheck` – не проверять права;
- `LCAccessTypes.CheckLCOnly` – проверять права жизненного цикла;
- `LCAccessTypes.CheckAll` – проверять права жизненного цикла и объекта;
- `LCAccessTypes.CheckInherited` – публиковать права входящим объектам.

`IsFirstStep`

Свойство определяет, является ли данный шаг первым в схеме жизненного цикла.

Синтаксис:

```
bool IsFirstStep { get; set; }
```

LCName

Наименование этапа ЖЦ.

Синтаксис:

```
string LCName{ get; set; }
```

LCStep

Идентификатор этапа ЖЦ (только для чтения).

Синтаксис:

```
int LCStep{ get; }
```

LevelID

Идентификатор уровня продвижения.

Синтаксис:

```
int LevelID{ get; set; }
```

Note

Комментарии.

Синтаксис:

```
string Note{ get; set; }
```

ObjectModifyMode

Способ модификации объектов на данном шаге жизненного цикла.

Синтаксис:

```
ObjectModifyModes ObjectModifyMode{ get; set; }
```

Способы модификации объектов:

- ObjectModifyModes.InBase – непосредственно в базе данных;
- ObjectModifyModes.CheckOut – через рабочую копию объекта;
- ObjectModifyModes.CreateVersion – выпуском новой версии объекта;
- ObjectModifyModes.CantModify – нельзя модифицировать объект.

ObjectTypeID

Идентификатор типа объекта, к которому относится данный этап жизненного цикла (только для чтения).

Синтаксис:

```
int ObjectTypeID{ get; }
```

Properties

Структура со свойствами этапа ЖЦ.

Синтаксис:

```
DBLifecycleStepProperties Properties{ get; set; }
```

AutoTransferStepID

Возвращает идентификатор шага ЖЦ, на который будут вытесняться версии с данного шага (или 0).

Синтаксис:

```
int AutoTransferStepID { get; }
```

IsDeleted

Возвращает true если шаг удалён.

Синтаксис:

```
bool IsDeleted { get; }
```

SchemaID

Идентификатор схемы, к которой принадлежит данный шаг жизненного цикла.

Синтаксис:

```
int SchemaID { get; }
```

3.8.2.3 Методы интерфейса

Delete

Удалить этап.

Синтаксис:

```
int Delete(Int64 DeleteMode);
```

Параметры:

DeleteMode

Определяет параметры удаления.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

GetDeleteStepID

Возвращает идентификатор шага, который доступен для объекта, находящегося на данном шаге, и имеет уровень продвижения "Удаленный".

Синтаксис:

```
int GetDeleteStepID();
```

Возвращаемое значение:

Идентификатор доступного для объекта шага жизненного цикла.

GetNextSteps

Метод получает идентификаторы шагов, на которые может быть перемещен объект, находящийся на данном шаге жизненного цикла.

Синтаксис:

```
int[] GetNextSteps();
```

Возвращаемое значение:

Идентификаторы шагов жизненного цикла, которые доступны для объекта на данном шаге.

SetGUID

Метод присваивает глобальный идентификатор шагу жизненного цикла.

Синтаксис:

```
void SetGUID(Guid guid).
```

GetAttributeSecurity

Возвращает интерфейс для проверки и назначения прав доступа на атрибут attrID на данном шаге ЖЦ применительно к типу объекта.

Синтаксис:

```
IDBSecurity GetAttributeSecurity(int attrID);
```

Параметры:

attrID - идентификатор атрибута.

Возвращаемое значение:

Интерфейс для работы с правами доступа.

3.9 Типы связей

Главная характеристика любой связи между информационными объектами, которая определяет:

- допустимый набор атрибутов, которые могут присваиваться связи автоматически или вручную;
- допустимые типы объектов, между которыми может существовать данная связь;
- хранение истории связей;
- способы обработки связей различными модулями системы и т.д.

Пространство имен: Intermech.Interfaces.

3.9.1 RelationTypeProperties

Структура, содержащая свойства типа связей.

3.9.1.1 Свойства

AnyAttributes

Свойство контролирует набор атрибутов: если принимает значение *false*, то допускается добавлять к объектам данного типа только разрешенные атрибуты, если *true* - допускается добавлять любые атрибуты.

Синтаксис:

```
public bool AnyAttributes;
```

AreaID

Строка символов, определяющих предметные области объектов, объединенных данной связью

Синтаксис:

```
public string AreaID;
```

CheckoutFile

Свойство определяет, нужно ли извлекать на диск файлы дочерних объектов, объединённых данной связью.

Синтаксис:

```
public bool CheckoutFile;
```

Description

Уникальное описание типа связи (например, Проектная связь)

Синтаксис:

```
public string Description;
```

Note

Комментарии.

Синтаксис:

```
public string Note;
```

RelationType

Идентификатор типа связи (только для чтения).

Синтаксис:

```
public int RelationType;
```

ReverseName

Наименование типа связи с точки зрения дочернего объекта (например, Входит в...).

Синтаксис:

```
public string ReverseName;
```

ShortName

Краткое наименование типа связи (может быть пустым).

Синтаксис:

```
public string ShortName;
```

TypeName

Наименование типа связи с точки зрения родительского объекта (например, Состоит из...).

Синтаксис:

```
public string TypeName.
```

Options

Опции типа связей.

Синтаксис:

```
public RelationTypeOptions Options;
```

SaveHistory

Нужно ли сохранять историю изменения связей в рамках одной версии.

Синтаксис:

```
public bool SaveHistory;
```

3.9.2 IDBRelationTypeCollection

Интерфейс позволяет создавать новые типы связей, получать список уже существующих типов связей, проверять существование типов с указанными идентификаторами и пр. Реализация осуществляется объектом `IDBRelationTypeCollection`.

3.9.2.1 Способы получения интерфейса

Для получения интерфейса используются функции:

- IUserSession. GetRelationType(Int64 aRelationTypeID);
- IUserSession. GetRelationType(Guid relationTypeGUID, bool throwException);
- IUserSession. GetRelationType(Guid relationTypeGUID);
- IUserSession. GetRelationType(string rtypeDescription);
- IUserSession. GetRelationType(string rtypeDescription, bool throwException);
- IUserSession. GetRelationType(Int64 projectID, Int64 partID);

Параметры:

aRelationTypeID

Идентификатор связи.

relationTypeGUID

Глобальный идентификатор связи.

throwException

Параметр определяет, генерировать ли исключение

rtypeDescription

Уникальное описание типа связи.

projectID

Идентификатор версии объекта-родителя.

partID

Идентификатор дочернего объекта.

3.9.2.2 Методы интерфейса

Create

Создает новый тип связей и возвращает его идентификатор.

Синтаксис:

```
int Create(RelationTypeProperties relationProperties);
```

Параметры:

relationProperties

Структура, содержащая свойства типа связей (описание см. п. [3.9.1](#)).

GetUsedByAttribute

Возвращает список типов связей, для которых задано использование атрибута номер attributeID.

Синтаксис:

```
DataTable GetUsedByAttribute(int attributeID);
```

Параметры:

attributeID

Идентификатор атрибута.

Select

Метод возвращает таблицу со списком всех типов связей, зарегистрированных в системе.

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка “F_TYPE_NAME DESC” отсортирует список типов связей по их наименованию в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_RELATION_TYPES.

addInfo

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица со списком типов связей.

Таблица IMS_RELATION_TYPES. Список типов связей.

Имя поля	Тип поля	Комментарий	IDBRelationType
F_RELATION_TYPE	INTEGER	Идентификатор типа связи.	RelationType
F_DESCRIPTION	NVARCHAR(255)	Наименование связи (например, Входямость пользователей в группы).	Description
F_TYPE_NAME	NVARCHAR(255)	Имя вида связи (например, Состоит из...).	TypeName
F_REVERSE_NAME	NVARCHAR(255)	Обратное название связи (например, для Состоит... из это Входит в...).	ReverseName
F_NOTE	NVARCHAR(450)	Комментарии.	Note
F_CHKOUTFILE	SMALLINT	Нужно ли извлекать на диск файлы объектов, объединённых данной связью (1 - нужно, 0 - не нужно).	CheckoutFile
F_ICON	LARGE BINARY	Иконка для отображения типа связей.	Icon
F_SAVE_HISTORY	SMALLINT	Данное поле не используется системой.	
F_GUID	VARCHAR(40)	Глобальный идентификатор типа связи (для портфеля и привязки обработчиков объектов).	IDBGuid.GUID
F_AREA_ID	NVARCHAR(8)	Предметная область.	IDBSubjectArea. SubjectAreas
F_ANY_ATTRIBUTES	SMALLINT	Контроль набора атрибутов: 0 - допускается добавлять к связям данного типа только разрешенные атрибуты; 1 - допускается добавлять любые атрибуты.	AnyAttributes
F_SHORT_NAME	NVARCHAR(32)	Краткое наименование типа связи.	ShortName

3.9.3 Attribute4RelationTypeProperties

Структура, описывающая свойства атрибута в контексте определенного типа связи.

AttributeID

Идентификатор атрибута (только для чтения).

Синтаксис:

```
public int AttributeID;
```

ComputeValueMode

Способ вычисления параметра.

Синтаксис:

```
public ComputeValueModes ComputeValueMode;
```

DefaultValue

Значение по умолчанию.

Синтаксис:

```
public object DefaultValue;
```

FieldType

Тип атрибута (строковый, числовой, т.д.).

Синтаксис:

```
public FieldTypes FieldType;
```

Formula

Формула вычисления значения поля. Для ссылок на объекты содержит номер атрибута, значение которого будет показываться методом AsString атрибута.

Синтаксис:

```
public string Formula;
```

IsContent

Свойство определяет, хранит ли атрибут содержимое объекта. Если у данного объекта есть атрибут, содержащий дату модификации, то изменение этого атрибута влияет на дату модификации объекта.

Синтаксис:

```
public bool IsContent;
```

Mask

Маска ввода значения атрибута.

Синтаксис:

```
public string Mask;
```

MasterAttributeID

Идентификатор мастер-атрибута для данного атрибута.

Синтаксис:

```
public int MasterAttributeID;
```

OptimizationMode

Задаёт способ оптимизации операций с атрибутом.

Синтаксис:

```
public OptimizationModes OptimizationMode;
```

Options

Опции атрибута (см. описание AttributeOptions).

Синтаксис:

```
public AttributeOptions Options;
```

RelationType

Тип связи атрибутов.

Синтаксис:

```
public int RelationType;
```

RequiredMode

Свойство описывает допустимость и обязательность атрибута для данного типа объектов и связей.

Синтаксис:

```
public RequiredModes RequiredMode;
```

SourceAttributeID

Идентификатор атрибута, из которого данный атрибут будет выбирать данные при присвоении значения мастер-атрибуту.

Синтаксис:

```
public int SourceAttributeID;
```

ValidationRule

Условие правильности вводимых в атрибут значений.

Синтаксис:

```
public string ValidationRule.
```

3.9.4 IDBAttribute4RelationTypeCollection

Интерфейс позволяет получать коллекцию атрибутов, допустимых для данного типа связей, создавать новые атрибуты, получать список уже существующих атрибутов для данного типа связей. Реализация осуществляется объектом IDBAttribute4RelationTypeCollection.

3.9.4.1 Способы получения интерфейса

Чтобы получить информацию о допустимости связи определенного типа между объектами дочернего и родительского типов, необходимо привести интерфейс реализующего объекта к интерфейсу IDBAttributableType, например:

```
UserSession.GetObjectType(typeID).Attributes as IDBAttribute4RelationTypeCollection;  
UserSession.GetObjectType(typeID).VisibleAttributes as IDBAttribute4RelationTypeCollection.
```

3.9.4.2 Методы интерфейса

Create

Метод создает атрибут и добавляет его к списку атрибутов типа связей.

Синтаксис:

IDBAttributeType4Relation Create(Attribute4RelationTypeProperties attrProperties);

Параметры:

attrProperties

Структура, описывающая свойства атрибута в контексте определенного типа связи (см. п. [3.9.3](#))

Select

Метод возвращает таблицу значений параметров связей между объектами.

Синтаксис:

DataTable Select (string orderBy, params object[] addInfo);

Параметры:

orderBy

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов ASC (сортировка по возрастанию) и DESC (сортировка по убыванию). Например, строка **F_RELATION_TYPE DESC** отсортирует список параметров по идентификатору типа связи в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице IMS_ATTR4RELATION_TYPES.

addInfo

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица значений параметров связей между объектами.

Таблица IMS_ATTR4RELATION_TYPES. Набор параметров для типа связи.

Имя поля	Тип поля	Комментарий	IDBRelationType
F_RELATION_TYPE	INTEGER	Идентификатор типа связи.	RelationType
F_ATTRIBUTE_ID	INTEGER	Идентификатор атрибута.	AttributeID
F_REQUIRED	SMALLINT	Допустимость и обязательность атрибута для данного типа связей.	RequiredMode
F_VALIDATION_RULE	NVARCHAR(450)	Некое условие правильности значений, попадающих в этот атрибут в контексте данного типа связи.	ValidationRule
F_COMPUTED	SMALLINT	Является ли параметр вычисляемым.	ComputeValueMode
F_FORMULA	NVARCHAR(450)	Формула вычисления значения.	Formula
F_DEFAULT_VALUE	NVARCHAR(450)	Значение по умолчанию.	DefaultValue
F_INVIEW	SMALLINT	Способ оптимизации записи/чтения атрибутов в общей таблице со списком объектов.	OptimizationMode
F_CONTENT	SMALLINT	Влияет ли данный атрибут на содержимое объекта (0 - нет, 1 - да).	IsContent
F_OPTIONS	INTEGER	Флаги с настройками параметров.	Options
F_MASK	NVARCHAR(450)	Маска ввода значения атрибута.	Mask
F_MASTER_ID	INTEGER	Идентификатор мастер-атрибута. Если идентификатор больше 0, то это подчиненный атрибут (его значение зависит от значения мастер-атрибута).	MasterAttributeID
F_SOURCE_ID	INTEGER	Идентификатор атрибута, из которого подчиненный атрибут выбирает данные у объекта-источника.	SourceAttributeID

3.9.5 IDBAttributeType4Relation

Интерфейс, описывающий свойства атрибута в контексте определенного типа связей. Реализуется объектом DBAttributeType4Relation. Интерфейс наследует все свойства и методы класса IDBAttributeType.

3.9.5.1 Свойства интерфейса

Attribute4RelationPropertiesStructure

Структура, описывающая свойства атрибута в контексте определенного типа связи (описание см. п. [3.9.3](#)).

Синтаксис:

```
Attribute4RelationTypeProperties Attribute4RelationPropertiesStructure { get; set; }
```

Required

Свойство описывает допустимость и обязательность атрибута для данного типа объектов и связей.

Синтаксис:

```
RequiredModes Required { get; set; }
```

Наличие атрибута у объектов данного типа:

- RequiredModes.Manual – атрибут может быть добавлен вручную;
- RequiredModes.Auto – необязательный атрибут, добавляемый автоматически;
- RequiredModes.AutoRequired – обязательный атрибут, добавляемый автоматически.

3.9.6 IDBRelationType

Интерфейс для работы с типом связи.

3.9.6.1 Свойства интерфейса

CheckoutFile

Свойство определяет, нужно ли извлекать на диск файлы объектов, объединённых данной связью.

Синтаксис:

```
bool CheckoutFile { get; set; }
```

Description

Уникальное описание типа связи (например, Проектная связь).

Синтаксис:

```
string Description { get; set; }
```

Icon

Иконка для отображения типа связей.

Синтаксис:

```
byte[] Icon { get; set; }
```

Note

Комментарии.

Синтаксис:

```
string Note { get; set; }
```

PropertiesStructure

Структура, позволяющая прочитать или записать сразу все параметры типа связей (описание см. п. [3.9.1](#)).

RelationType

Идентификатор типа связи (только для чтения).

Синтаксис:

```
int RelationType { get; }
```

ReverseName

Наименование типа связи с точки зрения дочернего объекта (например, Входит в...).

Синтаксис:

```
string ReverseName { get; set; }
```

ShortName

Краткое наименование типа связи (может быть пустым).

Синтаксис:

```
string ShortName { get; set; }
```

TypeName

Наименование типа связи с точки зрения родительского объекта (например, состоит из...).

Синтаксис:

```
string TypeName { get; set; }
```

3.9.6.2 Методы интерфейса

Delete

Метод удаляет тип связи из системы. Генерирует исключение `KernelException` в случае, если данный тип где-либо используется.

Синтаксис:

```
int Delete(Int64 DeleteMode);
```

Параметры:

`DeleteMode`

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

RebuildView

Пересоздает таблицу-представление данных для этого типа связей.

Синтаксис:

```
void RebuildView().
```

3.10 Допустимые связи между объектами

Связь между объектами – это категория, устанавливающая подчиненное отношение между версией информационного объекта и другим информационным объектом, т.е. в версию объекта с идентификатором `ObjectID` входит объект с идентификатором `ID`. Такой способ связывания позволяет оставлять без изменения связи родительского объекта при выпуске новой версии любого из дочерних объектов. Конкретизация версий дочерних объектов при получении состава родительского объекта производится с помощью текущего правила подбора версий. Все связи делятся на типы, которые определяют набор допустимых атрибутов у связи, видимость связей в клиентских приложениях, хранение истории связей и т.д.

Пространство имен: `Intermech.Interfaces`.

3.10.1 RelationsApplicabilityProperties

Структура, описывающая свойства контекста связи типа `RelationType` между объектами типов `ObjectType` (дочерний) и `InObjectType` (родительский).

ApplicabilityID

Идентификатор контекста связи.

Синтаксис:

```
public int ApplicabilityID;
```

ApplicabilityMode

Свойство обязательности связи.

Синтаксис:

```
public ApplicabilityModes ApplicabilityMode;
```

CheckoutFiles

Свойство определяет, нужно ли извлекать на диск файлы объектов, объединённых данной связью.

Синтаксис:

```
public bool CheckoutFiles;
```

CloneChildRelations

Свойство определяет, нужно ли при создании версии родительского объекта создавать версии дочерних объектов.

Синтаксис:

```
public bool CloneChildRelations;
```

InObjectType

Тип родительского объекта.

Синтаксис:

```
public int InObjectType;
```

IsContent

Свойство определяет, влияет ли данная связь на содержимое родительского объекта. Если значение свойства `IsContent` равно `true`, то при модификации атрибутов этой связи меняется дата модификации родительского объекта.

При создании объекта по прототипу родительского объекта копируются те связи, у которых значение свойства IsContent равно true.

Синтаксис:

```
public bool IsContent;
```

MaximumLinks

Максимальное количество применений объектов дочернего типа в объектах родительского типа связями relationType. Если значение свойства равно Int32.MaxValue, то допустимо бесконечно много связей.

Синтаксис:

```
public int MaximumLinks;
```

ObjectType

Тип дочернего объекта.

Синтаксис:

```
public int ObjectType;
```

Options

Опции связи.

Синтаксис:

```
public ApplicabilityOptions Options;
```

RelationConstraintMode

Способ обработки удаления объектов, объединенных этой связью.

Синтаксис:

```
public RelationConstraintModes RelationConstraintMode;
```

RelationType

Тип связи.

Синтаксис:

```
public int RelationType.
```

3.10.2 IDBRelationsApplicabilityCollection

Коллекция допустимых связей объектов данного типа с объектами различных типов. Реализуется классом DBRelationsApplicabilityCollection.

3.10.2.1 Методы интерфейса

Create

Создает новый контекст связи на основании структуры applicabilityProperties и возвращает его идентификатор.

Синтаксис:

```
int Create(RelationsApplicabilityProperties applicabilityProperties);
```

Параметры:

applicabilityProperties

Структура, описывающая свойства контекста связи типа RelationType между объектами типов ObjectType (дочерний) и InObjectType (родительский) (описание структуры см. п. [3.10.1](#)).

GetApplicabilitiesList

Возвращает таблицу со списком допустимых применений объектов типа objectType в объектах типа inObjectType связями типа relationType.

Если relationType меньше 0, то таблица содержит список типов связей, которыми объекты типа objectType могут входить в состав объектов типа inObjectType.

Если objectType меньше 0, то таблица содержит список применений всех типов в типе inObjectType.

Если inObjectType меньше 0, то таблица содержит список типов входимостей объектов типа objectType.

Если допустимых применений нет, то возвращается пустая таблица.

Синтаксис:

```
DataTable GetApplicabilitiesList ( int relationType, int objectType, int inObjectType);
```

Возвращаемое значение:

Таблица со списком допустимых применяемостей объектов типа objectType IMS_TYPES_APPLICABILITY. Описание таблицы приведено в методе Select.

GetApplicability

Возвращает объект-описатель допустимости связи типа `relationType` между объектами типов `objectType` и `inObjectType`. Если такое применение связи не настроено, то метод возвращает `null`.

Синтаксис:

```
IDBRelationsApplicability GetApplicability(int relationType, int objectType, int inObjectType);
```

Параметры:

`relationType`

Тип связи.

`objectType`

Тип дочернего объекта.

`inObjectType`

Тип родительского объекта.

GetApplicability

Возвращает объект, описывающий контекст связи типа `RelationType` между объектами типа `ObjectType` (дочерний) и `InObjectType` (родительский) допустимости связи с идентификатором `applicabilityID`.

Синтаксис:

```
IDBRelationsApplicability GetApplicability(int applicabilityID);
```

Параметры:

`applicabilityID`

Идентификатор контекста связи.

GetApplicabilityRow

Возвращает строку таблицы `IMS_TYPES_APPLICABILITY` допустимости связи типа `relationType` между объектами типов `objectType` и `inObjectType`.

Если таковая применяемость не настроена, то метод возвращает `null`.

Синтаксис:

```
DataRow GetApplicabilityRow(int relationType, int objectType, int inObjectType)
```

Параметры:

`relationType`

Тип связи.

`objectType`

Тип дочернего объекта.

`inObjectType`

Тип родительского объекта.

Select

Метод возвращает таблицу допустимых применяемостей типов объектов.

Синтаксис:

```
DataTable Select (string orderBy, params object[] addInfo);
```

Параметры:

`orderBy`

Имена полей, по которым необходимо отсортировать записи в возвращаемой таблице. Порядок сортировки указывается с помощью ключевых слов `ASC` (сортировка по возрастанию) и `DESC` (сортировка по убыванию). Например, строка **F_RELATION_TYPE DESC** отсортирует список типов применяемостей связей по идентификатору типа связи в убывающем порядке. Если ключевое слово не указывать, то сортировка производится по возрастанию значений поля. Допустимые имена полей см. в таблице `IMS_TYPES_APPLICABILITY`.

`addInfo`

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Таблица допустимых применяемостей типов объектов.

Таблица `IMS_TYPES_APPLICABILITY`. Список применяемости связей.

Имя поля	Тип поля	Комментарий	IDBRelationApplicability
F_APPLICABILITY_ID	INTEGER	Идентификатор контекста типа связи.	ApplicabilityID
F_INOBJECT_TYPE	INTEGER	Идентификатор типа родительского объекта.	InObjectType

F_OBJECT_TYPE	INTEGER	Идентификатор типа дочернего объекта.	ObjectType
F_RELATION_TYPE	INTEGER	Идентификатор типа связи.	RelationType
F_MAX_LINKS	INTEGER	Максимально допустимое количество применений объектов дочернего типа в объектах родительского типа связями relationType.	MaximumLinks
F_MIN_LINKS	INTEGER	Минимально допустимое количество применений данной связи.	ApplicabilityMode
F_CLONE_RELATIONS	SMALLINT	Нужно ли при создании версии родительского объекта F_INOBJECT_TYPE создавать версию дочернего объекта F_OBJECT_TYPE.	CloneChildRelations
F_CONSTRAINT_MODE	SMALLINT	Способ обработки удаления объектов, объединенных этой связью.	RelationConstraintMode
F_CHKOUTFILE	SMALLINT	Нужно ли извлекать на диск файлы объектов, объединённых данной связью	CheckoutFiles
F_PUBLIC	SMALLINT	Используется ядром для определения принадлежности связи (своя или унаследованная).	-
F_CONTENT	SMALLINT	Влияет ли данная связь на содержимое объекта.	IsContent
F_OPTIONS	INTEGER	Опции атрибута (набор флагов).	Options

3.10.3 IDBRelationsApplicability

Интерфейс объекта, описывающего контекст связи типа RelationType между объектами типов ObjectType (дочерний) и InObjectType (родительский).

3.10.3.1 Свойства интерфейса

ApplicabilityID

Идентификатор контекста связи.

Синтаксис:

```
int ApplicabilityID { get; }
```

ApplicabilityMode

Свойство обязательности связи.

Синтаксис:

```
ApplicabilityModes ApplicabilityMode { get; set; }
```

Контроль существования связи:

- ApplicabilityModes.Disabled – запрещенная связь;
- ApplicabilityModes.Enabled – разрешенная связь;
- ApplicabilityModes.Required – обязательная связь;
- ApplicabilityModes.AnyRequired – одна из обязательных связей.

CloneChildRelations

Если значение свойства true, то при создании версии родительского объекта нужно создавать версии дочерних объектов.

Синтаксис:

```
bool CloneChildRelations { get; set; }
```

InObjectType

Тип родительского объекта.

Синтаксис:

```
int InObjectType { get; }
```

IsContent

Свойство определяет, влияет ли данная связь на содержимое родительского объекта. Если значение свойства IsContent равно *true*, то при модификации атрибутов этой связи меняется дата модификации родительского объекта.

При создании объекта по прототипу родительского объекта копируются те связи, у которых значение свойства IsContent равно *true*.

Синтаксис:

```
bool IsContent { get; set; }
```

MaximumLinks

Максимальное количество таких связей. Если свойство принимает значение *Int32.MaxValue*, то допустимо бесконечно много связей.

Синтаксис:

```
int MaximumLinks { get; set; }
```

ObjectType

Тип дочернего объекта.

Синтаксис:

```
int ObjectType { get; }
```

Options

Опции связи.

Синтаксис:

```
ApplicabilityOptions Options { get; set; }
```

RelationConstraintMode

Способ обработки удаления объектов, объединенных этой связью.

Синтаксис:

```
RelationConstraintModes RelationConstraintMode { get; set; }
```

Способы обработки удаления связанных объектов:

- *RelationConstraintModes.None* – запрещенная связь;
- *RelationConstraintModes.ChildConstrained* – запрещать удаление дочернего объекта;
- *RelationConstraintModes.ParentConstrained* – запрещать удаление родительского объекта;
- *RelationConstraintModes.ParentChildConstrained* – запрещать удаление родительского и дочернего объектов;
- *RelationConstraintModes.ChildDelete* – удалять дочерний объект при удалении родительского объекта;
- *RelationConstraintModes.ParentDelete* – удалять родительский объект при удалении всех дочерних объектов, включенных в его состав связью данного типа.

RelationType

Тип связи.

Синтаксис:

```
int RelationType { get; }
```

3.10.3.2 Методы интерфейса

Delete

Метод удаляет контекст связи.

Синтаксис:

```
int Delete();
```

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

3.11 Дополнительный кэш метаданных (класс `MetaDataHelper`)

Для получения информации о типах объектов, связей, атрибутов, а также о применяемых можно воспользоваться системным кэшем метаданных, либо обращаться к пользовательской сессии, получая у неё описания типов объектов и т.д. (ссылки на интерфейсы `IDBObjectType`, `IDBRelationType`, `IDBAttributeType`, `IDBRelationsApplicability`). Но если требуется выполнять данные действия многократно, следует использовать дополнительный кэш метаданных, который реализован в статическом классе `MetaDataHelper`.

Статический класс `MetaDataHelper` доступен на серверной и клиентской сторонах, содержит упорядоченные коллекции классов, описывающих системные метаданные. Для того, чтобы данный класс получил из системного кэша метаданных всю необходимую информацию, у него есть ряд методов для синхронизации (`SyncMetaData` и аналогичные). Кэш позволяет выполнять полную или частичную синхронизацию, в зависимости от того, в какой части метаданных произошли изменения. Например, если был добавлен новый тип связи, достаточно синхронизировать часть дополнительного кэша, связанную с типами связей.

Рассмотрим подробно методы и свойства, позволяющие поддерживать дополнительный кэш в актуальном состоянии – методы синхронизации с системным кэшем метаданных.

3.11.1 Пространство имён

`Intermech.Interfaces`

Статический класс, доступный как на серверной стороне, так и на клиентской. Позволяет быстро получать из системного кэша метаданных информацию о типах объектов, связей и атрибутов, об иерархии типов объектов, применяемости типов связей и типов объектов в составах.

3.11.2 Свойства класса

`SyncDelta`

Интервал времени, в течение которого не будет выполняться синхронизация внутренних полей класса с системным кэшем метаданных после последней синхронизации. Например, если вызывать методы по синхронизации с интервалом времени меньшим, чем `SyncDelta`, то эти дополнительные вызовы будут проигнорированы.

Синтаксис:

```
public static TimeSpan SyncDelta { get; }
```

3.11.3 Методы класса

Рассмотрим методы, позволяющие выполнить синхронизацию всех или только некоторых внутренних коллекций с системным кэшем метаданных.

`SyncMetadata`

Метод выполняет полную синхронизацию внутренних коллекций с системным кэшем метаданных. При этом выполняется ряд проверок, чтобы не получать из кэша информацию, если в ней не было изменений. Метод вызывает в определённом порядке остальные методы, синхронизирующие внутренние коллекции с кэшем. Сам метод вызывается при инициализации серверного ядра, а также при инициализации клиентского ядра. Вручную вызывать его следует, только если есть необходимость, например, были внесены изменения в системные метаданные.

Синтаксис:

```
void SyncMetadata(IUserSession session);
```

Параметры:

`session`

Пользовательская сессия, в рамках которой выполняется синхронизация с системным кэшем метаданных.

`SyncObjectTypesMetadata`

Метод выполняет синхронизацию внутренних коллекций, связанных с типами объектов, с системным кэшем метаданных. При этом выполняется ряд проверок, чтобы не получать из кэша информацию, если в ней не было изменений.

Синтаксис:

```
void SyncObjectTypesMetadata (IUserSession session);
```

Параметры:

`session`

Пользовательская сессия, в рамках которой выполняется синхронизация с системным кэшем метаданных.

`SyncObjectTypesHierarchy`

Метод выполняет синхронизацию внутренних коллекций, связанных с иерархией типов объектов, с системным кэшем метаданных. При этом выполняется ряд проверок, чтобы не получать из кэша информацию, если в ней не было изменений.

Синтаксис:

```
void SyncObjectTypesHierarchy (IUserSession session);
```

Параметры:

session

Пользовательская сессия, в рамках которой выполняется синхронизация с системным кэшем метаданных.

SyncRelationTypesMetadata

Метод выполняет синхронизацию внутренних коллекций, связанных с коллекцией типов связей, с системным кэшем метаданных. При этом выполняется ряд проверок, чтобы не получать из кэша информацию, если в ней не было изменений.

Синтаксис:

```
void SyncRelationTypesMetadata (IUserSession session);
```

Параметры:

session

Пользовательская сессия, в рамках которой выполняется синхронизация с системным кэшем метаданных.

SyncSpecialObjectTypes

Метод выполняет синхронизацию внутренних коллекций, связанных со специальными (системными) типами объектов, с системным кэшем метаданных. При этом выполняется ряд проверок, чтобы не получать из кэша информацию, если в ней не было изменений.

Синтаксис:

```
void SyncSpecialObjectTypes (IUserSession session);
```

Параметры:

session

Пользовательская сессия, в рамках которой выполняется синхронизация с системным кэшем метаданных.

SyncSpecialRelationTypes

Метод выполняет синхронизацию внутренних коллекций, связанных со специальными (системными) типами связей, с системным кэшем метаданных. При этом выполняется ряд проверок, чтобы не получать из кэша информацию, если в ней не было изменений.

Синтаксис:

```
void SyncSpecialRelationTypes (IUserSession session);
```

Параметры:

session

Пользовательская сессия, в рамках которой выполняется синхронизация с системным кэшем метаданных.

SyncAttrTypesMetadata

Метод выполняет синхронизацию внутренних коллекций, связанных с коллекцией атрибутов в системе, с системным кэшем метаданных. При этом выполняется ряд проверок, чтобы не получать из кэша информацию, если в ней не было изменений.

Синтаксис:

```
void SyncAttrTypesMetadata (IUserSession session);
```

Параметры:

session

Пользовательская сессия, в рамках которой выполняется синхронизация с системным кэшем метаданных.

Рассмотрим подробно функционал дополнительного кэша метаданных.

3.11.4 Получение информации о типах объектов

Данная часть кэша помогает получать описания типов объектов по их Int32 или Guid идентификаторам, отыскать названия типов объектов, выполнять преобразования Int32-идентификаторов типов объектов в их Guid идентификаторы и наоборот.

3.11.4.1 Методы класса

GetObjectType

Метод возвращает описание указанного типа объекта по его Int32 или Guid идентификатору.

Синтаксис:

- `IMSObjectType GetObjectType (Int32 objTypeID);`
- `IMSObjectType GetObjectType (Guid objTypeGuid);`

Параметры:

`objTypeID, objTypeGuid`

`Int32` или `Guid` идентификатор искомого типа объекта.

Результат:

Экземпляр класса **IMSObjectType**, поля которого содержат описание типа объекта, либо **null**, если указанное описание типа объекта не найдено.

GetObjectTypeName

Метод возвращает название типа объектов (например, «Детали») по его `Int32` или `Guid` идентификатору.

Синтаксис:

`string GetObjectTypeName (Int32 objTypeID);`

`string GetObjectTypeName (Guid objTypeGuid);`

Параметры:

`objTypeID, objTypeGuid`

`Int32` или `Guid` идентификатор искомого типа объекта.

Результат:

Название типа объектов.

GetObjectName

Метод возвращает название экземпляра объектов (например, «Деталь») по `Int32` или `Guid` идентификатору его типа.

Синтаксис:

`string GetObjectName (Int32 objTypeID);`

`string GetObjectName (Guid objTypeGuid);`

Параметры:

`objTypeID, objTypeGuid`

`Int32` или `Guid` идентификатор искомого типа объекта.

Результат:

Название экземпляра объекта указанного типа.

ExistsObjectType

Метод проверяет существование указанного типа объектов по `Int32` или `Guid` идентификатору его типа.

Синтаксис:

`bool ExistsObjectType (Int32 objTypeID);`

`bool ExistsObjectType (Guid objTypeGuid);`

Параметры:

`objTypeID, objTypeGuid`

`Int32` или `Guid` идентификатор искомого типа объекта.

Результат:

true, если указанный тип объекта найден в кэше метаданных.

GetObjectTypeID

Метод возвращает `Int32`-идентификатор типа объекта по его `Guid` идентификатору или по строковому представлению его `Guid` идентификатора.

Синтаксис:

`Int32 GetObjectTypeID (Guid objTypeGuid);`

`Int32 GetObjectTypeID (string Guid);`

Параметры:

`objTypeGuid`

`Guid` идентификатор искомого типа объекта.

`Guid`

Строковое представление `Guid` идентификатора искомого типа объекта.

Результат:

Int32-идентификатор типа объекта. Значение **-1** означает то, что указанный тип объекта не найден.

GetObjectTypeGuid

Метод возвращает Guid идентификатор типа объекта по его Int32-идентификатору.

Синтаксис:

```
Guid GetObjectTypeGuid (Int32 objTypeID);
```

Параметры:

objTypeID

Int32-идентификатор искомого типа объекта.

Результат:

Guid идентификатор типа объекта. Значение **Guid.Empty** означает то, что указанный тип объекта не найден.

3.11.5 Получение информации о типах связей

Данная часть кэша помогает получать описания типов связей по их Int32 или Guid идентификаторам, отыскивать названия типов связей, выполнять преобразования Int32-идентификаторов типов связей в их Guid идентификаторы и наоборот.

3.11.5.1 Методы класса

GetRelationType

Метод возвращает описание указанного типа связи по его Int32 или Guid идентификатору.

Синтаксис:

```
IMSRelationType GetRelationType (Int32 relTypeID);
```

```
IMSRelationType GetRelationType (Guid relTypeGuid);
```

Параметры:

relTypeID, relTypeGuid

Int32 или Guid идентификатор искомого типа связи.

Результат:

Экземпляр класса IMSRelationType, поля которого содержат описание типа связи, либо null, если указанное описание типа связи не найдено.

GetRelationTypeName

Метод возвращает название указанного типа связи по его Int32 или Guid идентификатору.

Синтаксис:

```
string GetRelationTypeName (Int32 relTypeID);
```

```
string GetRelationTypeName (Guid relTypeGuid);
```

Параметры:

relTypeID, relTypeGuid

Int32 или Guid идентификатор искомого типа связи.

Результат:

Название указанного типа связи или **string.Empty**, если тип связи не найден.

ExistsRelationType

Метод проверяет, существует ли указанный тип связи в кэше метаданных.

Синтаксис:

```
bool ExistsRelationType (Int32 relTypeID);
```

```
bool ExistsRelationType (Guid relTypeGuid);
```

Параметры:

relTypeID, relTypeGuid

Int32 или Guid идентификатор искомого типа связи.

Результат:

true, если указанный тип связи найден в кэше метаданных.

GetRelationTypeID

Метод возвращает Int32-идентификатор типа связи по его Guid идентификатору или по строковому представлению его Guid идентификатора.

Синтаксис:

```
Int32 GetRelationTypeID (Guid relTypeGuid);
```

```
Int32 GetRelationTypeID (string Guid);
```

Параметры:

relTypeGuid

Guid идентификатор искомого типа связи.

Guid

Строковое представление Guid идентификатора искомого типа связи.

Результат:

Int32-идентификатор типа связи. Значение **-1** означает то, что указанный тип связи не найден.

GetRelationTypeGuid

Метод возвращает Guid идентификатор типа связи по его Int32-идентификатору.

Синтаксис:

```
Guid GetRelationTypeGuid (Int32 relTypeID);
```

Параметры:

relTypeID

Int32-идентификатор искомого типа связи.

Результат:

Guid идентификатор типа связи. Значение **Guid.Empty** означает то, что указанный тип связи не найден.

GetRelationType4PrjLinkID

Метод позволяет получить из кэша или из базы данных тип указанной связи. Если задать значение session равное null, то значение будет получено из кэша. Если в кэше такого значения нет, будет возвращено значение – 1. Данный метод интенсивно используется основной частью универсального клиента – **Навигатором**.

Синтаксис:

```
Int32 GetRelationType4PrjLinkID (IUserSession session, Int64 prjLinkID);
```

Параметры:

session

Сессия, в рамках которой выполняется работа с базой данных.

prjLinkID

Идентификатор связи, тип которой требуется определить.

Результат:

Идентификатор типа связи или - 1.

3.11.6 Получение информации об иерархии типов объектов

Данная часть кэша помогает получать информацию о наследовании типов объектов, списки дочерних и родительских типов объектов.

3.11.6.1 Методы класса

GetObjectTypeParentsID

Метод возвращает список Int32-идентификаторов всех родительских типов объектов для указанного дочернего типа объекта. Например, тип [A] – базовый, от него унаследован тип [B], а от типа [B] унаследован тип [C]. Данный метод для типа [C] вернёт список идентификаторов его родительских типов объектов [B, A].

Синтаксис:

```
List<Int32> GetObjectTypeParentsID (Int32 childTypeID);
```

```
List<Int32> GetObjectTypeParentsID (Guid childTypeGuid);
```

Параметры:

childTypeID

Int32-идентификатор дочернего типа объекта.

childTypeGuid

Guid идентификатор дочернего типа объекта.

Результат:

Список Int32-идентификаторов всех родительских типов объектов для указанного дочернего типа объекта.

GetObjectTypeParentsGuid

Метод возвращает список Guid идентификаторов всех родительских типов объектов для указанного дочернего типа объекта. Работает метод аналогично методу GetObjectTypeParentsID.

Синтаксис:

```
List<Guid> GetObjectTypeParentsGuid (Int32 childTypeID);
```



```
List<Guid> GetObjectTypeParentsGuid (Guid childTypeGuid);
```

Параметры:

childTypeID

Int32-идентификатор дочернего типа объекта.

childTypeGuid

Guid идентификатор дочернего типа объекта.

Результат:

Список Guid идентификаторов всех родительских типов объектов для указанного дочернего типа объекта.

GetObjectTypeChildrenID

Метод возвращает список Int32-идентификаторов всех дочерних типов объектов для указанного родительского типа объекта на один уровень ниже. Например, от базового типа [A] унаследованы типы [B] и [V]. Метод для типа [A] вернёт список [B, V]. Типы объектов, унаследованные от [B] и/или [V], метод в данном случае не возвращает.

Синтаксис:

```
List<Int32> GetObjectTypeChildrenID (Int32 parentTypeID);
```

```
List<Int32> GetObjectTypeChildrenID (Guid parentTypeGuid);
```

Параметры:

parentTypeID

Int32-идентификатор родительского типа объекта.

parentTypeGuid

Guid идентификатор родительского типа объекта.

Результат:

Список Int32-идентификаторов всех дочерних типов объектов для указанного родительского типа объекта.

GetObjectTypeChildrenGuid

Метод возвращает список Guid идентификаторов всех дочерних типов объектов для указанного родительского типа объекта. Метод работает аналогично методу GetObjectTypeChildrenID.

Синтаксис:

```
List<Guid> GetObjectTypeChildrenGuid (Int32 parentTypeID);
```

```
List<Guid> GetObjectTypeChildrenGuid (Guid parentTypeGuid);
```

Параметры:

parentTypeID

Int32-идентификатор родительского типа объекта.

parentTypeGuid

Guid идентификатор родительского типа объекта.

Результат:

Список Guid идентификаторов всех дочерних типов объектов для указанного родительского типа объекта.

3.11.7 Получение информации о типах связей по умолчанию

Данная часть кэша помогает получать идентификаторы типов связей по умолчанию для указанных типов объектов.

3.11.7.1 Методы класса

GetDefaultRelationTypeID

Метод возвращает Int32-идентификатор типа связи по умолчанию для указанного родительского типа объекта.

Синтаксис:

```
Int32 GetDefaultRelationTypeID (Int32 parentObjectType);
```

```
Int32 GetDefaultRelationTypeID (Guid parentObjectTypeGuid);
```

Параметры:

parentObjectType

Int32-идентификатор родительского типа объекта.

parentObjectTypeGuid

Guid идентификатор родительского типа объекта.

Результат:

Int32-идентификатор типа связи по умолчанию для указанного типа объектов, или **-1**, если тип объекта не найден.

GetDefaultRelationTypeGuid

Метод возвращает Guid идентификатор типа связи по умолчанию для указанного родительского типа объекта.

Синтаксис:

```
Guid GetDefaultRelationTypeGuid (Int32 parentObjectType);  
Guid GetDefaultRelationTypeGuid (Guid parentObjectTypeGuid);
```

Параметры:

parentObjectType

Int32-идентификатор родительского типа объекта.

parentObjectTypeGuid

Guid идентификатор родительского типа объекта.

Результат:

Guid идентификатор типа связи по умолчанию для указанного типа объектов, или **Guid.Empty**, если тип объекта не найден.

3.11.8 Получение информации о допустимых типах связей

Данная часть кэша помогает получать списки допустимых типов связей для указанных типов объектов.

3.11.8.1 Методы класса

GetApplicabilityRelationTypesID

Метод возвращает список Int32-идентификаторов допустимых типов связей для указанного родительского типа объекта.

Синтаксис:

```
List<Int32> GetApplicabilityRelationTypesID (Int32 objTypeID);  
List<Int32> GetApplicabilityRelationTypesID (Guid objTypeGuid);
```

Параметры:

objTypeID

Int32-идентификатор родительского типа объекта.

objTypeGuid

Guid идентификатор родительского типа объекта.

Результат:

Список Int32-идентификаторов допустимых типов связей для указанного родительского типа объекта.

GetApplicabilityRelationTypesGuids

Метод возвращает список Guid идентификаторов допустимых типов связей для указанного родительского типа объекта.

Синтаксис:

```
List<Guid> GetApplicabilityRelationTypesGuids (Int32 objTypeID);  
List<Guid> GetApplicabilityRelationTypesGuids (Guid objTypeGuid);
```

Параметры:

objTypeID

Int32-идентификатор родительского типа объекта.

objTypeGuid

Guid идентификатор родительского типа объекта.

Результат:

Список Guid идентификаторов допустимых типов связей для указанного родительского типа объекта.

3.11.9 Получение информации о применяемости

Данная часть кэша помогает получать списки применяемостей (дочерние типы объектов и допустимые типы связей для указанных родительских типов объектов), а также списки дочерних типов объектов для указанных родительских типов объектов и допустимых типов связей.

3.11.9.1 Методы класса

GetObjectTypeApplicabilities

Метод возвращает список применяемостей (допустимый тип связи + дочерний тип объекта) для указанного родительского типа объекта.

Синтаксис:

```
List<IMSApplicability> GetObjectTypeApplicabilities (Int32 objTypeID);  
List<IMSApplicability> GetObjectTypeApplicabilities (Guid objTypeGuid);
```

Параметры:

objTypeID

Int32-идентификатор родительского типа объекта.

objTypeGuid

Guid идентификатор родительского типа объекта.

Результат:

Список применяемостей (допустимый тип связи + дочерний тип объекта) для указанного родительского типа объекта.

GetApplicabilityChildObjectTypes

Метод возвращает список дочерних типов объектов, которые можно включать в состав указанного родительского типа объекта по указанному типу связи.

Синтаксис:

```
List<IMSObjectType> GetApplicabilityChildObjectTypes (Int32 parObjTypeID, Int32 relTypeID);  
List<IMSObjectType> GetApplicabilityChildObjectTypes (Guid parObjTypeGuid, Guid relTypeGuid);
```

Параметры:

parTypeID

Int32-идентификатор родительского типа объекта.

parObjTypeGuid

Guid идентификатор родительского типа объекта.

relTypeID

Int32-идентификатор типа связи.

relTypeGuid

Guid идентификатор типа связи.

Результат:

Список дочерних типов объектов, которые можно включать в состав указанного родительского типа объекта по указанному типу связи.

GetApplicabilityChildObjectTypesID

Метод возвращает список Int32-идентификаторов дочерних типов объектов, которые можно включать в состав указанного родительского типа объекта по указанному типу связи.

Синтаксис:

```
List<Int32> GetApplicabilityChildObjectTypesID (Int32 parObjTypeID, Int32 relTypeID);  
List<Int32> GetApplicabilityChildObjectTypesID (Guid parObjTypeGuid, Guid relTypeGuid);
```

Параметры:

parTypeID

Int32-идентификатор родительского типа объекта.

parObjTypeGuid

Guid идентификатор родительского типа объекта.

relTypeID

Int32-идентификатор типа связи.

relTypeGuid

Guid идентификатор типа связи.

Результат:

Список Int32-идентификаторов дочерних типов объектов, которые можно включать в состав указанного родительского типа объекта по указанному типу связи.

GetApplicabilityChildObjectTypesGuid

Метод возвращает список Guid идентификаторов дочерних типов объектов, которые можно включать в состав указанного родительского типа объекта по указанному типу связи.

Синтаксис:

```
List<Guid> GetApplicabilityChildObjectTypesGuid (Int32 parObjTypeID, Int32 relTypeID);
```

```
List<Guid> GetApplicabilityChildObjectTypesGuid (Guid parObjTypeGuid, Guid relTypeGuid);
```

Параметры:

parTypeID

Int32-идентификатор родительского типа объекта.

parObjTypeGuid

Guid идентификатор родительского типа объекта.

relTypeID

Int32-идентификатор типа связи.

relTypeGuid

Guid идентификатор типа связи.

Результат:

Список Guid идентификаторов дочерних типов объектов, которые можно включать в состав указанного родительского типа объекта по указанному типу связи.

3.11.10 Получение информации о специальных типах объектов

Данная часть кэша помогает получать описания некоторых специальных (системных) типов объектов.

3.11.10.1 Методы класса

HasObjectTypeSubstRelTypes

Метод проверяет, может ли указанный родительский тип объекта содержать в своём составе связи, участвующие в допустимых заменах.

Синтаксис:

```
bool HasObjectTypeSubstRelTypes (Int32 objTypeID);
```

```
bool HasObjectTypeSubstRelTypes (Guid objTypeGuid);
```

Параметры:

objTypeID

Int32-идентификатор типа объекта.

objTypeGuid

Guid идентификатор типа объекта.

Результат:

true, если указанный родительский тип объекта может содержать в своём составе связи, участвующие в допустимых заменах.

HasObjectTypeSortingRelTypes

Метод проверяет, может ли указанный родительский тип объекта содержать в своём составе связи, позволяющие выполнять ручную сортировку.

Синтаксис:

```
bool HasObjectTypeSortingRelTypes (Int32 objTypeID);
```

```
bool HasObjectTypeSortingRelTypes (Guid objTypeGuid);
```

Параметры:

objTypeID

Int32-идентификатор типа объекта.

objTypeGuid

Guid идентификатор типа объекта.

Результат:

true, если указанный родительский тип объекта может содержать в своём составе связи, позволяющие выполнять ручную сортировку.

HasObjectTypeDesignedRelType

Метод проверяет, может ли указанный родительский тип объекта содержать в своём составе связи типа **Состав изделия**.

Синтаксис:

```
bool HasObjectTypeDesignedRelType (Int32 objTypeID);
```

```
bool HasObjectTypeDesignedRelType (Guid objTypeGuid);
```

Параметры:

objTypeID

Int32-идентификатор типа объекта.

objTypeGuid

Guid идентификатор типа объекта.

Результат:

true, если указанный родительский тип объекта может содержать в своём составе связи типа **Состав изделия**.

HasObjectTypeGroupingRelType

Метод проверяет, является ли указанный тип объектов группирующим, и при этом может ли содержать в своём составе группирующие типы связей.

Синтаксис:

```
bool HasObjectTypeGroupingRelType (Int32 objTypeID);
```

```
bool HasObjectTypeGroupingRelType (Guid objTypeGuid);
```

Параметры:

objTypeID

Int32-идентификатор типа объекта.

objTypeGuid

Guid идентификатор типа объекта.

Результат:

true, если указанный родительский тип объекта является группирующим, и при этом в его состав могут входить группирующие типы связей.

GetSpecialGroupingIDs

Метод возвращает список Int32-идентификаторов группирующих типов объектов.

Синтаксис:

```
List<Int32> GetSpecialGroupingIDs ();
```

Результат:

Список Int32-идентификаторов группирующих типов объектов.

GetSpecialGroupingGuids

Метод возвращает список Guid идентификаторов группирующих типов объектов.

Синтаксис:

```
List<Guid> GetSpecialGroupingGuids ();
```

Результат:

Список Guid идентификаторов группирующих типов объектов.

3.11.11 Получение информации о специальных типах связей

Данная часть кэша помогает получать описания некоторых специальных (системных) типов связей.

3.11.11.1 Методы класса

HasRelationTypeSorting

Метод проверяет, поддерживает ли указанный тип связи работу с ручной сортировкой.

Синтаксис:

```
bool HasRelationTypeSorting (Int32 relTypeID);
```

```
bool HasRelationTypeSorting (Guid relTypeGuid);
```

Параметры:

relTypeID

Int32-идентификатор типа связи.

relTypeGuid

Guid идентификатор типа связи.

Результат:

true, если указанный тип связи поддерживает работу с ручной сортировкой.

HasRelationTypeSubstitutes

Метод проверяет, поддерживает ли указанный тип связи работу с допустимыми заменами.

Синтаксис:

```
bool HasRelationTypeSubstitutes (Int32 relTypeID);
```

```
bool HasRelationTypeSubstitutes (Guid relTypeGuid);
```

Параметры:

relTypeID

Int32-идентификатор типа связи.

relTypeGuid

Guid идентификатор типа связи.

Результат:

true, если указанный тип связи поддерживает работу с допустимыми заменами.

HasRelationTypeGrouping

Метод проверяет, поддерживает ли указанный тип связи группирование объектов (содержит в своём составе атрибуты **Идентификатор версии в составе, Сортировка**).

Синтаксис:

```
bool HasRelationTypeGrouping (Int32 relTypeID);
```

```
bool HasRelationTypeGrouping (Guid relTypeGuid);
```

Параметры:

relTypeID

Int32-идентификатор типа связи.

relTypeGuid

Guid идентификатор типа связи.

Результат:

true, если указанный тип связи поддерживает группирование объектов.

3.11.12 Получение информации о типах атрибутов

Данная часть кэша помогает получать описания типов атрибутов по их Int32 или Guid идентификаторам, отыскиать названия типов атрибутов, выполнять преобразования Int32-идентификаторов типов атрибутов в их Guid идентификаторы и наоборот.

3.11.12.1 Методы класса

GetAttributeType

Метод позволяет получить описание типа атрибутов.

Синтаксис:

```
IMSAttributeType GetAttributeType (Int32 attrTypeID);
```

```
IMSAttributeType GetAttributeType (Guid attrTypeGuid);
```

Параметры:

attrTypeID

Int32-идентификатор типа атрибута.

attrTypeGuid

Guid идентификатор типа атрибута.

Результат:

Описание типа атрибута или **null**, если указанный тип атрибута не найден в кэше метаданных.

GetAttributeName

Метод позволяет получить описание типа атрибутов.

Синтаксис:

```
string GetAttributeName (Int32 attrTypeID);
```

```
string GetAttributeName (Guid attrTypeGuid);
```

Параметры:

attrTypeID

Int32-идентификатор типа атрибута.

attrTypeGuid

Guid идентификатор типа атрибута.

Результат:

Название типа атрибута или string.Empty, если указанный тип атрибута не найден в кэше метаданных.

ExistsAttributeType

Метод позволяет проверить существование указанного типа атрибута в кэше метаданных.

Синтаксис:

```
bool ExistsAttributeType (Int32 attrTypeID);  
bool ExistsAttributeType (Guid attrTypeGuid);
```

Параметры:

attrTypeID

Int32-идентификатор типа атрибута.

attrTypeGuid

Guid идентификатор типа атрибута.

Результат:

true, если указанный тип атрибута существует в кэше метаданных.

HasAttributeSystemData

Метод позволяет проверить, хранятся ли в указанном атрибуте системные данные (тип данных у атрибута равен FieldTypes.ftSystem).

Синтаксис:

```
bool HasAttributeSystemData (Int32 attrTypeID);  
bool HasAttributeSystemData (Guid attrTypeGuid);
```

Параметры:

attrTypeID

Int32-идентификатор типа атрибута.

attrTypeGuid

Guid идентификатор типа атрибута.

Результат:

true, если указанный тип атрибута хранит системные данные (тип данных у атрибута равен FieldTypes.ftSystem).

HasAttributePossibleValues

Метод позволяет проверить, является ли атрибут списковым.

Синтаксис:

```
bool HasAttributePossibleValues (Int32 attrTypeID);  
bool HasAttributePossibleValues (Guid attrTypeGuid);
```

Параметры:

attrTypeID

Int32-идентификатор типа атрибута.

attrTypeGuid

Guid идентификатор типа атрибута.

Результат:

true, если указанный тип атрибута является списковым.

IsAttributeGridable

Метод позволяет проверить, можно ли отображать значения указанного типа атрибута в открытом виде (например, атрибуты с паролем так отображать нельзя).

Синтаксис:

```
bool IsAttributeGridable (Int32 attrTypeID);  
bool IsAttributeGridable (Guid attrTypeGuid);
```

Параметры:

attrTypeID

Int32-идентификатор типа атрибута.

attrTypeGuid

Guid идентификатор типа атрибута.

Результат:

true, если можно отображать значения указанного типа атрибута в открытом виде.

GetAttributeTypeID

Метод позволяет получить Int32-идентификатор типа атрибута по его Guid идентификатору или строковому представлению его Guid идентификатора.

Синтаксис:

```
Int32 GetAttributeTypeID (Guid attrTypeGuid);
```

```
Int32 GetAttributeTypeID (string Guid);
```

Параметры:

attrTypeGuid

Guid идентификатор типа атрибута.

Guid

Строковое представление Guid идентификатора типа атрибута.

Результат:

Int32-идентификатор типа атрибута или – 1, если указанный тип атрибута не найден.

GetAttributeTypeGuid

Метод позволяет получить Guid идентификатор типа атрибута по его Int32-идентификатору.

Синтаксис:

```
Guid GetAttributeTypeGuid (Int32 attrTypeID);
```

Параметры:

attrTypeID

Int32-идентификатор типа атрибута.

Результат:

Guid идентификатор типа атрибута или **Guid.Empty**, если указанный тип атрибута не найден.

GetAttributeTypesList

Метод позволяет получить список описаний всех типов атрибутов из системного кэша метаданных.

Синтаксис:

```
List<IMSAttributeType> GetAttributeTypesList ();
```

Результат:

Список описаний всех типов атрибутов из системного кэша метаданных.

3.11.13 Класс IMSObjectType – описание типа объекта

Экземпляры данного класса хранят в себе описания типов объектов, взятые из системной таблицы IMS_OBJECT_TYPES. Они используются в дополнительном кэше метаданных MetadataHelper. Вручную создавать экземпляры этого типа не требуется.

3.11.13.1 Пространство имён

Intermech.Interfaces

Поля экземпляра данного класса хранят описание типа объекта из кэша метаданных.

3.11.13.2 Поля класса

Int32 ObjectTypeID

Идентификатор типа объекта.

Guid Guid

Guid идентификатор типа объекта.

string ObjectTypeName

Наименование типа объектов (например, **Детали**).

string ObjectName

Наименование объекта данного типа (например, **Деталь**).

ObjectVersionModes VersionsMode

Управление версионностью типов объектов.

Int32 DefaultRelation

Идентификатор типа связи по умолчанию.

string AreaID

Предметная область.

Int32 SchemaID

Идентификатор схемы жизненного цикла.

string Note

Примечание

ObjectTypeOptions Options

Опции, регулирующие поведение типов объектов.

Int32 CaptionAttribute

Атрибут-описатель.

bool AnyAttributes

Можно ли назначить любые типы атрибутов экземплярам данного типа.

string ShortName

Краткое наименование.

3.11.14 Класс IMSRelationType – описание типа связи

Экземпляры данного класса хранят в себе описания типов связей, взятые из системной таблицы IMS_RELATION_TYPES. Они используются в дополнительном кэше метаданных MetaDataHelper. Вручную создавать экземпляры этого типа не требуется.

3.11.14.1 Пространство имён

Intermech.Interfaces

Поля экземпляра данного класса хранят описание типа связи из кэша метаданных.

3.11.14.2 Поля класса

Int32 RelationTypeID

Идентификатор типа связи.

Guid Guid

Guid идентификатор типа связи.

string Description

Наименование типа связи (например, **Состав изделия**).

string TypeName

Имя вида связи (например, **Состоит из**).

string ReverseName

Обратное название связи (например, **Входит в**).

Int32 ChkOutFile

Нужно ли извлекать на диск файлы объектов, объединённых данной связью (1 – нужно, 0 – не нужно).

Int32 RelationKind

Вид связи: 0 – вертикальная (например, состоит из); 1 – горизонтальная (взаимозаменяемый).

string AreaID

Предметная область.

string Note

Примечание.

bool AnyAttributes

Можно ли назначать связям данного типа любые атрибуты.

string ShortName

Краткое название.

RelationTypeOptions Options

Опции (содержат битовые флаги для управления свойствами типа связей).

3.11.15 Класс IMSAttributeType – описание типа атрибута

Экземпляры данного класса хранят в себе описания типов атрибутов, взятые из системной таблицы IMS_ATTRIBUTES. Они используются в дополнительном кэше метаданных MetaDataHelper. Вручную создавать экземпляры этого типа не требуется.

3.11.15.1 Пространство имён

Intermech.Interfaces

Поля экземпляра данного класса хранят описание типа атрибута из кэша метаданных.

3.11.15.2 Поля класса

Int32 AttributeID

Идентификатор типа атрибута.

string Name

Название типа атрибута.

string ShortName

Краткое название типа атрибута.

string Alias

Альтернативное имя атрибута (для хранения понятий Техкарда).

string Note

Примечание к типу атрибута.

FieldTypes FieldType

Тип данных атрибута (строковый, числовой, т.д.).

FieldTypes RealFieldType

Реальный тип данных атрибута (если FieldType == FieldTypes.ftSystem).

object DefaultValue

Значение по умолчанию.

MultiValueModes MultiValueMode

Может ли атрибут принимать множественные значения и каким образом.

ComputeValueModes Computed

Способ вычисления параметра.

Int64 SizeType

Размер данных, хранящихся в атрибуте:

1. для строковых параметров - максимальная длина строки,
2. для ссылки на объект – идентификатор типа объекта,
3. для таблицы из внешней БД – ссылка на объект, описывающий эту БД, таблицу и поля с ключом, и значением.

string Formula

Формула вычисления значения поля. Для ссылок на объекты содержит номер атрибута, значение которого будет показываться методом AsString атрибута.

UniqueValueModes Unique

Метод контроля уникальности значений атрибута.

Int32 LevelID

Идентификатор уровня продвижения, на котором можно применять данный атрибут.

string LanguageID

Идентификатор языка.

string AreaID

Идентификатор предметной области.

OptimizationModes OptimizationMode

Способ оптимизации операций с атрибутом.

bool IsContent

Влияет ли изменение значения атрибута на дату модификации объекта (если таковая имеется).

AttributeOptions Options

Опции атрибута.

string Mask

Маска ввода значения атрибута.

Int32 MasterAttributeID

Идентификатор мастер-атрибута для данного атрибута.

Int32 SourceAttributeID

Идентификатор атрибута, из которого данный атрибут будет выбирать данные при присвоении значения мастер-атрибуту.

3.11.15.3 Свойства класса

bool IsGridable { get; }

Можно ли отображать значение указанного типа атрибута в открытом виде.

3.11.16 Класс IMSApplicability – описание применяемости

Экземпляры данного класса хранят в себе описания применяемостей, взятые из системной таблицы IMS_TYPES_APPLICABILITY. Они используются в дополнительном кэше метаданных MetaDataHelper. Вручную создавать экземпляры этого типа не требуется.

3.11.16.1 Пространство имён

Intermech.Interfaces

Поля экземпляра данного класса хранят описание применяемости (допустимого дочернего типа объекта, который можно добавить в состав указанного родительского типа объекта по указанному типу связи).

3.11.16.2 Поля класса

Int32 ApplicabilityID

Идентификатор контекста связи (применяемости).

Int32 ChildObjectTypeID

Идентификатор дочернего типа объектов.

Int32 InObjectType

Идентификатор родительского типа объектов.

Int32 RelationTypeID

Идентификатор допустимого типа связи между указанными типами объектов.

bool CloneChildRelations

Если true, то при создании версии родительского объекта нужно копировать данные связи у исходной версии объекта или у объекта-прототипа.

bool CheckoutFiles

Извлекать ли на диск файлы по таким типам связей.

Int32 MaximumLinks

Максимально допустимое количество таких связей. Если = Int32.MaxValue, то бесконечное.

RelationConstraintModes RelationConstraintMode

Способ обработки удаления объектов, связанных этой связью.

ApplicabilityModes ApplicabilityMode

Свойство обязательности связи.

bool IsContent

Влияет ли данная связь на содержимое родительского объекта. Если да, то при модификации атрибутов этой связи меняется дата модификации родительского объекта. Также при создании объекта по прототипу родительского объекта копируются все связи, у которых IsContent == true.

ApplicabilityOptions Options

Опции связи.

InheritModes Public

Используется для определения наследования типа связи (свой тип связи или унаследованный).

4 Атрибуты объектов и связей

4.1 Базовые перечислители, структуры, классы

4.1.1 Перечислитель GetAttributeValuesModes

Битовый перечислитель ([Flags()]) описывает режимы формирования массива структур значений атрибутов – **AttributeValues** в функции **IDBAttributable.GetAttributesValues** для управления массивом возвращаемой информации.

4.1.1.1 Пространство имён

Intermech.Interfaces

4.1.1.2 Значения перечислителя

None = 0

Зарезервировано.

IncludeName = 1

Заполняет поле **AttributeName** в **AttributeValues**.

IncludeGuid = 2

Заполняет поле **AttributeGuid**.

IncludeAlias = 4

Заполняет поле **AttributeAlias**.

IncludeBlobs = 8

Включает в возвращаемый список атрибутов атрибуты типа **ftBlob**, **ftShortBlob** и **ftFile**.

IncludeObligatoryAttributes = 16

Включает в возвращаемый список атрибутов обязательные атрибуты (**ftSystem**).

IncludeGroupName = 32

Заполняет поле **GroupName** именем первой из групп атрибутов, куда включен данный атрибут.

CheckWriteAccess = 64

Заполняет поле **ReadOnly**, осуществляя проверку возможности модификации каждого атрибута. При этом анализируются права доступа, статус объекта и т.п., поэтому данный флаг не рекомендуется включать без необходимости.

IncludeDescriptions = 128

Заполняет поле **Descriptions** для атрибутов, значение которых могут потребовать строковой расшифровки.

CheckVisibility = 256

Включает в список только атрибуты, видимые в интерфейсе данному пользователю.

IncludeOnlyInvisible = 512

Включать в список только атрибуты, невидимые в интерфейсе данному пользователю. Позволяет дочитать список атрибутов объекта, уже зачитанный с флагом **CheckVisibility**, чтобы получить полный набор атрибутов объекта.

IncludeCaption = 1024

Включает в список атрибутов заголовков объекта (применяется, если не нужны остальные обязательные атрибуты).

4.1.2 Структура AttributeDataTableValue

Структура для хранения всех составляющих значения атрибута.

4.1.2.1 Пространство имён

Intermech.Interfaces

4.1.2.2 Свойства структуры

DateValue

Дата и время.

Синтаксис:

public DateTime DateValue;

DoubleValue

Вещественная часть.

Синтаксис:
public double DoubleValue;
IntegerValue
Целочисленная часть.
Синтаксис:
public Int64 IntegerValue;
StringValue
Строковая часть.
Синтаксис:
public string StringValue;
InlistID
Индекс значения.
Синтаксис:
public int InlistID;

4.2 Интерфейсы для работы со списками атрибутов

4.2.1 IDBAttributable

Базовый интерфейс для работы с экземплярами атрибутов. Поддерживается информационными объектами и связями. Интерфейс унаследован от интерфейса **IDBSessionable**.

4.2.1.1 Пространство имён

Intermech.Interfaces

4.2.1.2 Свойства интерфейса

Attributes

Список атрибутов. Первое обращение к данному свойству приводит к чтению из базы данных всех атрибутов объекта/связи (кроме обязательных атрибутов **FieldTypes.ftSystem**). Если требуется работать только с одним атрибутом, то рекомендуется использовать методы получения атрибута у самого интерфейса **IDBAttributable** (например, **IDBAttributable.GetAttributeByID**).

Синтаксис:
IDBAttributeCollection Attributes { get; }

ReadOnly

Значение **true** данного свойства говорит о том, что данный объект (связь) нельзя редактировать. Однако существуют ситуации, когда отдельные атрибуты редактироваться всё равно могут (например, атрибуты, у которых в конфигураторе для данного типа объектов включен флаг **Возможна модификация без взятия на изменение**). Также возможна и обратная ситуация – объект (связь) изменять можно, а на какой-либо атрибут(ы) правами доступа назначен запрет редактирования. В связи с этим рекомендуется проверять возможность редактирования у самого атрибута **IDBAttribute.ReadOnly**.

Синтаксис:
bool ReadOnly { get; }

TypeID

Возвращает тип объекта или связи, которому (которой) принадлежат эти атрибуты.

Синтаксис:
Int32 TypeID { get; }

4.2.1.3 Методы интерфейса

GetAttributeByGuid

Метод возвращает обработчик для атрибута по глобальному идентификатору атрибута. Если у объекта нет такого атрибута, то возвращается **null**.

Синтаксис:
IDBAttribute GetAttributeByGuid(Guid attributeGuid);
IDBAttribute GetAttributeByGuid(Guid attributeGuid, bool throwNotFoundException);

Параметры:
attributeGuid

Глобальный идентификатор атрибута.

`throwNotFoundException`

Значение параметра определяет, будет ли генерироваться исключение **AttributeNotFoundException** в случае отсутствия такого атрибута. В реализации без данного параметра исключение не генерируется.

Результат:

Обработчик для атрибута или **null**.

GetAttributeByID

Метод возвращает обработчик атрибута по его локальному идентификатору. Если у объекта (связи) нет такого атрибута, то возвращается **null**.

Синтаксис:

```
IDBAttribute GetAttributeByID(int attributeID);
```

Параметры:

`attributeID`

Идентификатор атрибута.

Результат:

Обработчик для атрибута или **null**.

GetAttributeByName

Метод возвращает обработчик атрибута по его наименованию. Если у объекта нет такого атрибута, то возвращается **null**.

Синтаксис:

```
IDBAttribute GetAttributeByName(string attributeName);
```

```
IDBAttribute GetAttributeByName(string attributeName, bool throwNotFoundException);
```

Параметры:

`attributeName`

Наименование атрибута.

`throwNotFoundException`

Значение параметра определяет, будет ли генерироваться исключение **AttributeNotFoundException** в случае отсутствия такого атрибута.

Результат:

Обработчик для атрибута или **null**.

GetAttributesValues

Возвращает массив значений атрибутов в соответствии с указанными в параметре `modes` настройками.

Синтаксис:

```
AttributeValues[] GetAttributesValues(GetAttributeValuesModes modes);
```

Параметры:

`modes`

Параметр определяет режимы формирования массива структур **AttributeValues**, которые содержат идентификаторы атрибутов и их значения. Режимы указываются в виде битовых флагов, значения которых описаны в п. [4.1.1](#).

GetValuesByGuid

Функция возвращает значения атрибута с глобальным идентификатором **guid**, если он существует. Если значение параметра **throwNotFoundException** равно **true**, а атрибута не существует, то возвращается **null**. Метод работает для всех атрибутов, включая обязательные.

Синтаксис:

```
object[] GetValuesByGuid(Guid guid, bool throwNotFoundException);
```

GetValuesByID

Возвращает значения атрибута с локальным идентификатором **attributeID**, если он существует. Если значение параметра **throwNotFoundException** равно **true**, а атрибута с данным идентификатором не существует, то возвращается **null**. Метод возвращает значения и обязательных атрибутов.

Синтаксис:

```
object[] GetValuesByID(int attributeID, bool throwNotFoundException);
```

Параметры:

`attributeID`

Локальный идентификатор атрибута.

throwNotFoundException

Значение параметра определяет, будет ли генерироваться исключение **AttributeNotFoundException** в случае отсутствия такого атрибута у данного объекта (связи).

SetAttributesValues

Метод записывает значения атрибутам объекта посредством массива **valuesList**. Недостающие атрибуты добавляются объекту (если это возможно). Обязательные атрибуты также могут быть записаны данным методом. Если значение свойства **valuesList[i].ReadOnly** равно **true**, то данный атрибут не записывается. Атрибуты для хранения двоичных данных и файлов также игнорируются.

Синтаксис:

```
AttributeValues[] SetAttributesValues(AttributeValues[] valuesList,  
bool deleteNotExistingAttributes, bool dontDeleteBlobs, bool returnDelta, GetAttributeValuesModes  
modes);
```

Параметры:

valuesList

Массив структур, которые содержат идентификаторы атрибутов и их значения.

deleteNotExistingAttributes

Если значение параметра равно **true**, то отсутствующие в данном списке атрибуты (кроме обязательных) будут удалены.

dontDeleteBlobs

Если значение параметра равно **true**, то атрибуты для хранения двоичных данных удаляться не будут.

returnDelta

Если значение параметра **returnDelta** равно **true**, то метод возвращает массив атрибутов, которые были изменены сервером в результате изменения атрибутов, указанных в **valuesList**. Например, будут возвращены значения вычисляемых атрибутов, значения которых изменились в результате присвоения новых значений атрибутам их **valuesList**. Если **returnDelta** равно **false**, то функция возвращает **null**.

modes

Параметр определяет режимы формирования массива измененных значений (если **returnDelta = true**).

SetAttributesValues

Метод и его параметры имеют тот же смысл, что и в предыдущем случае. Параметру **returnDelta** передается значение **false**, параметру **modes** – значение 0. Метод всегда возвращает **null**.

Синтаксис:

```
AttributeValues[] SetAttributesValues(AttributeValues[] valuesList,  
bool deleteNotExistingAttributes, bool dontDeleteBlobs);
```

SetAttributesValues

Метод и его параметры имеют тот же смысл, что и в предыдущем случае. Параметру **deleteNotExistingAttributes** передается значение **false**, параметру **dontDeleteBlobs** – значение **true**, **returnDelta** – значение **false**. Метод всегда возвращает **null**.

Синтаксис:

```
AttributeValues[] SetAttributesValues(AttributeValues[] valuesList);
```

GetAttributeType

Метод возвращает обработчик типа атрибута номер **attributeID** применительно к данному объекту/связи.

Синтаксис:

```
public IDBAttributeType GetAttributeType(int attributeID);
```

attributeID – идентификатор атрибута.

4.2.2 IDBAttributeCollection

Интерфейс позволяет получать коллекцию атрибутов для данного объекта или связи, добавлять новые атрибуты, получать список уже существующих атрибутов для данного объекта или связи. Для получения коллекции используется свойство **IDBAttributable.Attributes**. В коллекции отсутствуют обязательные атрибуты объектов (связей) – для чтения и модификации их значений используйте соответствующие методы интерфейсов **IDBAttributable**, **IDBObject** или **IDBRelation**.

4.2.2.1 Пространство имён

Intermech.Interfaces

4.2.2.2 Свойства интерфейса

Count

Возвращает количество атрибутов в списке.

Синтаксис:

```
int Count{ get; }
```

Item

Свойство возвращает атрибут с номером *i* в списке атрибутов.

Синтаксис:

```
IDBAttribute this[int AttrIndex]{ get; }
```

Параметры:

AttrIndex

Номер атрибута в списке.

ObjectID

Идентификатор версии объекта или связи, к которым относится данный список атрибутов (только для чтения).

Синтаксис:

```
Int64 ObjectID{ get; }
```

ObjectType

Идентификатор типа объекта или связи, к которым относится данный список атрибутов (только для чтения).

Синтаксис:

```
int ObjectType{ get; }
```

4.2.2.3 Методы интерфейса

AddAttribute

Метод добавляет атрибут с идентификатором attributeID к объекту (связи) и возвращает интерфейс обработчика добавленного атрибута.

Синтаксис:

```
IDBAttribute AddAttribute(int attributeID, bool failIfExists);
```

```
IDBAttribute AddAttribute(int attributeID, bool failIfExists, object[] initValues);
```

Параметры:

attributeID

Идентификатор атрибута.

failIfExists

Если значение параметра **failIfExists** равно **true** и атрибут уже существует, то генерируется исключение. Если значение **failIfExists** равно **false** и атрибут уже существует, то метод возвращает этот атрибут.

initValues

Значения, которыми инициализируется добавленный атрибут. Если параметр не указан, то атрибут инициализируется значением по умолчанию, заданным в конфигураторе БД.

AddDeltaValue

Добавляет идентификатор атрибута для формирования списка измененных сервером атрибутов.

Синтаксис:

```
void AddDeltaValue(int attributeID);
```

Параметры:

attributeID

Идентификатор атрибута.

AddTemporaryAttribute

Добавляет временный атрибут номер **attributeID** к объекту (связи) и инициализирует его значениями **initValues**. Временные атрибуты хранятся только в памяти, и для них не производится проверка допустимости их применения к данному типу объектов и связей. Временные атрибуты существуют в памяти столько, сколько существует обработчик объекта (связи), к которому они добавлены.

Синтаксис:

```
IDBAttribute AddTemporaryAttribute(int attributeID, bool failIfExists);
```

```
IDBAttribute AddTemporaryAttribute(int attributeID, bool failIfExists, object[] initValues);
```

Параметры:

attributeID

Идентификатор атрибута.

`failIfExists`

Если значение параметра **failIfExists** равно **true** и атрибут уже существует, то генерируется исключение. Если значение **failIfExists** равно **false** и атрибут уже существует, то метод возвращает этот атрибут.

Assign

Добавляет к данному списку атрибутов все атрибуты из коллекции **sourceAttributes**. Если в списке такой атрибут уже есть, то его значение меняется. Отсутствующие в **sourceAttributes** атрибуты не удаляются.

Синтаксис:

```
void Assign(IDBAttributeCollection sourceAttributes);
```

Параметры:

`sourceAttributes`

Список атрибутов объекта или связи.

ClearDeltaValues

Очищает список измененных сервером атрибутов. Вызывается автоматически после выполнения некоторых функций (например, **SetAttributesValues**).

Синтаксис:

```
void ClearDeltaValues();
```

FindByAlias

Возвращает атрибут по его псевдониму. Возвращает **null**, если атрибут в коллекции отсутствует.

Синтаксис:

```
IDBAttribute FindByAlias(string attributeAlias);
```

Параметры:

`attributeAlias`

Псевдоним атрибута.

FindByGUID

Возвращает атрибут по его глобальному идентификатору. Возвращает **null**, если атрибут в коллекции отсутствует.

Синтаксис:

```
IDBAttribute FindByGUID(Guid AttributeGUID);
```

Параметры:

`AttributeGUID`

Глобальный идентификатор атрибута.

FindByID

Возвращает атрибут по его локальному идентификатору. Возвращает **null**, если атрибут в коллекции отсутствует.

Синтаксис:

```
IDBAttribute FindByID(int AttributeID);
```

Параметры:

`AttributeID`

Идентификатор атрибута.

FindByName

Возвращает атрибут по его наименованию. Возвращает **null**, если атрибут в коллекции отсутствует.

Синтаксис:

```
IDBAttribute FindByName(string AttributeName);
```

Параметры:

`AttributeName`

Наименование атрибута.

GetAttributesByType

Возвращает список атрибутов заданного типа, которые есть у данного объекта (связи).

Синтаксис:

```
IDBAttribute[] GetAttributesByType(FieldTypes ft);
```

Параметры:

`ft`

Тип данных атрибута.

GetAttributesDataTable

Возвращает таблицу со значениями всех дополнительных атрибутов объекта/связи.

Синтаксис:

```
DataTable GetAttributesDataTable();
```

Возвращаемое значение:

Таблица IMS_OBJECT_ATTRS, если атрибут принадлежит объекту, или IMS_RELATION_ATTRS, если атрибут принадлежит связи.

Таблица IMS_OBJECT_ATTRS. Список значений атрибутов объекта.

Имя поля	Тип поля	Комментарий	IDBAttribute
F_ATTRIBUTE_ID	INTEGER	Идентификатор атрибута.	AttributeID
F_OBJECT_ID	BIGINT	Идентификатор версии объекта.	DBObjectID
F_INLIST_ID	INTEGER	Номер значения в списке значений.	Index
F_INTEGER_VALUE	BIGINT	Целочисленное значение.	AsInteger
F_STRING_VALUE	NVARCHAR(450)	Строковое значение.	AsString
F_DOUBLE_VALUE	FLOAT	Вещественное значение.	AsDouble
F_DATE_VALUE	DATETIME	Дата и время (в UTC).	AsDateTime

Таблица IMS_RELATION_ATTRS. Список значений атрибутов связи.

Имя поля	Тип поля	Комментарий	IDBAttribute
F_ATTRIBUTE_ID	INTEGER	Идентификатор атрибута.	AttributeID
F_PRJLINK_ID	BIGINT	Идентификатор записи.	DBObjectID
F_INLIST_ID	INTEGER	Номер значения в списке значений.	Index
F_INTEGER_VALUE	BIGINT	Целочисленное значение.	AsInteger
F_STRING_VALUE	NVARCHAR(450)	Строковое значение.	AsString
F_DOUBLE_VALUE	FLOAT	Вещественное значение.	AsDouble
F_DATE_VALUE	DATETIME	Дата и время (в UTC).	AsDateTime

GetDeltaValues

Метод возвращает список значений атрибутов, которые были изменены серверной частью системы (например, вычисляемых атрибутов или измененных плагинами значений). Если таковых нет, то возвращается **null**.

Синтаксис:

```
AttributeValues[] GetDeltaValues(GetAttributeValuesModes modes);
```

Параметры:

modes

Параметр определяет режимы формирования возвращаемого методом массива структур со значениями атрибутов. Описание управляющих флагов см. в п. [4.1.1](#). Данный метод не поддерживает флаги **IncludeObligatoryAttributes**, **CheckVisibility**, **IncludeOnlyInvisible**.

4.3 Работа с экземпляром атрибута

Экземпляром атрибута называется именованная характеристика, присвоенная информационному объекту или связи между объектами.

4.3.1 Класс AttributeValues

Класс, содержащий идентификатор(ы) атрибута и его значение(я).

4.3.1.1 Пространство имён

Intermech.Interfaces

4.3.1.2 Поля класса

AttributeAlias

Псевдоним атрибута.

Синтаксис:

```
public string AttributeAlias
```

AttributeGuid

Глобальный идентификатор атрибута.

Синтаксис:

```
public Guid AttributeGuid
```

AttributeID

Локальный идентификатор атрибута.

Синтаксис:

```
public Int32 AttributeID
```

AttributeName

Наименование атрибута.

Синтаксис:

```
public string AttributeName
```

AttributeType

Тип данных атрибута.

Синтаксис:

```
public FieldTypes AttributeType
```

ComputeMode

Тип вычисления атрибута.

Синтаксис:

```
public ComputeValueModes ComputeMode
```

Descriptions

Строковые расшифровки значений атрибута.

Синтаксис:

```
public object[] Descriptions
```

ReadOnly

Если значение равно **true**, то атрибут не может быть модифицирован в настоящий момент, иначе - зависит от состояния **CheckAccess** при получении **AttributeValues**.

Синтаксис:

```
public bool ReadOnly
```

ThrowSetException

Если значение атрибута равно **false**, то при невозможности записи данного атрибута в базу систем не генерирует исключение, что позволяет при пакетной записи не прерывать запись всех атрибутов в случае, если атрибут со значением **ThrowSetException** равно **false** записать не получается. По умолчанию содержит **true**.

Синтаксис:

```
public bool ThrowSetException
```

Values

Значения атрибута. Тип данных в массиве зависит от типа данных атрибута. Для двоичных и файловых атрибутов свойство всегда должно содержать **null**.

Синтаксис:

```
public object[] Values
```

4.3.2 Интерфейс IDBAttribute

Интерфейс для работы с атрибутом. Поддерживается всеми типами атрибутов, кроме обязательных (**FieldType.ftSystem**). При этом некоторые методы и свойства могут выдавать исключение в случае, если они не поддерживаются в данном типе атрибута. Особенности работы методов с разными типами атрибутов будут рассмотрены ниже.

4.3.2.1 Пространство имён

Intermech.Interfaces

4.3.2.2 Свойства интерфейса

AsBoolean

Значение атрибута как логическая величина.

Синтаксис:

```
bool AsBoolean { get; set; }
```

AsDateTime

Значение атрибута как дата и время (в локальном времени). При чтении генерирует исключение если атрибут содержит пустое значение **DBNull.Value**.

Синтаксис:

```
DateTime AsDateTime { get; set; }
```

AsDouble

Вещественное значение атрибута. При чтении генерирует исключение если атрибут содержит пустое значение.

Синтаксис:

```
double AsDouble { get; set; }
```

AsInteger

Целочисленное значение атрибута. При чтении генерирует исключение если атрибут содержит пустое значение.

Синтаксис:

```
int64 AsInteger { get; set; }
```

AsString

Строковое значение атрибута. Возвращает пустую строку если атрибут содержит пустое значение.

Синтаксис:

```
string AsString { get; set; }
```

AttributeID

Локальный идентификатор атрибута (только для чтения).

Синтаксис:

```
int AttributeID { get; }
```

AttributeType

Свойство возвращает интерфейс-описатель применимости атрибута в данном типе связи или объекта, т.е. его можно пробовать приводить к типам **IDBAttributeType4**, **IDBAttributeType4Object**, **IDBAttributeType4Relation** в зависимости от обстоятельств. Но следует учитывать, что в некоторых случаях он может оставаться только **IDBAttributeType** (если атрибут не был назначен в конфигураторе для данного типа объектов/связей).

Синтаксис:

```
IDBAttributeType AttributeType { get; }
```

DataType

Тип данных атрибута (только для чтения).

Синтаксис:

```
FieldTypes DataType { get; }
```

DBObjectID

Идентификатор версии объекта или связи, атрибутом которых он является (только для чтения).

Синтаксис:

```
int64 DBObjectID { get; }
```

Description

Возвращает расшифрованное текстовое представление текущего значения атрибута. Например, для ссылки на объект будет содержать заголовок объекта, на который ссылается атрибут.

Синтаксис:

```
string Description { get; }
```

Descriptions

Возвращает расшифрованное текстовое представление всех значений атрибута.

Синтаксис:

```
string[] Descriptions { get; }
```

GroupName

Возвращает имя первой из групп, в которые входит данный атрибут.

Синтаксис:

```
string GroupName { get; }
```

Index

Индекс текущего значения в списке значений. Может принимать значения от 0 до **ValuesCount** – 1. По умолчанию значение свойства равно 0.

Синтаксис:

```
int Index { get; set; }
```

IsNull

Свойство возвращает значение **true**, если атрибут содержит пустое значение.

Синтаксис:

```
bool IsNull { get; }
```

Name

Наименование атрибута (только для чтения).

Синтаксис:

```
string Name { get; }
```

ReadOnly

Свойство возвращает значение **true**, если данный пользователь не может модифицировать значение атрибута.

Синтаксис:

```
bool ReadOnly { get; }
```

TemporaryAttribute

Если значение свойства равно **true**, то данный атрибут временный и присутствует только в памяти.

Синтаксис:

```
bool TemporaryAttribute { get; }
```

Value

Свойство возвращает и модифицирует текущее значение атрибута (для двоичных и файловых атрибутов возвращает **null**).

Синтаксис:

```
object Value { get; set; }
```

Values

Возвращает или модифицирует список всех значений атрибута (для двоичных и файловых атрибутов возвращает **null**).

Синтаксис:

```
object[] Values { get; set; }
```

ValuesCount

Свойство возвращает количество значений в списке значений атрибута. Не может быть < 1, т.е. у атрибута всегда есть хотя бы одно значение.

Синтаксис:

```
int ValuesCount { get; }
```

Visible

Свойство возвращает значение **true**, если данный атрибут виден в пользовательском интерфейсе текущей сессии пользователя (по правам доступа и различным фильтрам).

Синтаксис:

```
bool Visible { get; }
```

4.3.2.3 Методы интерфейса

AddValue

Метод добавляет значение в список значений атрибута (для атрибутов, которые могут содержать множество значений) и возвращает номер добавленного значения в списке.

Синтаксис:

```
int AddValue(object newValue);
```

Параметры:

newValue

Если параметр содержит не пустое значение, то оно записывается в качестве добавляемого значения атрибута.

Результат:

Номер добавленного значения.

Assign

Копирует значение атрибута **sourceAttribute** в данный атрибут.

Синтаксис:

```
void Assign(IDBAttribute sourceAttribute);
```

Параметры:

sourceAttribute

Атрибут, значение которого копируется в данный атрибут.

Clear

Метод присваивает текущему значению атрибута пустое значение (если это допускается данным атрибутом).

Синтаксис:

```
void Clear();
```

ClearValues

Удаляет все значения многозначного атрибута, а последнее значение очищает.

Синтаксис:

```
void ClearValues();
```

Delete

Удалить атрибут.

Синтаксис:

```
int Delete(Int64 DeleteMode);
```

Параметры:

DeleteMode

Параметр используется для передачи в команду удаления различных управляющих флагов. В обычной ситуации в метод следует передавать 0.

Результат:

Всегда возвращает **0**. В случае каких-либо ошибок удаления метод генерирует исключения.

DeleteValue

Удаляет текущее значение из списка значений атрибута. Последнее значение удалить нельзя.

Синтаксис:

```
int DeleteValue();
```

Результат:

Всегда возвращает **0**. В случае каких-либо ошибок удаления метод генерирует исключения.

GetPossibleValues

Возвращает таблицу со списком значений, которые может принимать атрибут. В зависимости от типа данных атрибута в таблице будет заполнено одно из полей с соответствующим типом данных. Поле F_DESCRIPTION может не заполняться – в этом случае в качестве расшифровки значения будет выступать само значение.

Синтаксис:

```
DataTable GetPossibleValues();
```

Результат:

Таблица значений атрибута.

Имя поля	Тип поля	Комментарий	IDBAttribute
F_INLIST_ID	INTEGER	Номер записи в списковых параметрах.	Index
F_INTEGER_VALUE	BIGINT	Целочисленное значение.	AsInteger

F_STRING_VALUE	NVARCHAR(450)	Строковое значение.	AsString
F_DOUBLE_VALUE	FLOAT	Вещественное значение.	AsDouble
F_DATE_VALUE	DATETIME	Дата и время.	AsDateTime
F_DESCRIPTION	NVARCHAR(450)	Расшифрованное текстовое представление значения атрибута.	Description

4.4 Особенности работы с различными типами данных

4.4.1 Строки (ftString)

Атрибут, тип данных которого представляет собой строковые значения длиной до 450 символов, позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения-записи своего значения:

Использование свойств атрибута со строковым значением.

Свойство	Чтение	Запись
AsBoolean	Возвращает значение атрибута, конвертированное в логическое выражение. Выдает исключение, если конвертировать значение невозможно.	Записывает строку “Истина” или “Ложь” в зависимости от присваиваемого значения.
AsDateTime*	Возвращает значение атрибута, конвертированное в локальные дату и время. Выдает исключение, если конвертировать значение невозможно.	Конвертирует дату и время в строковое значение.
AsDouble*	Возвращает значение атрибута, конвертированное в вещественное число. Выдает исключение, если конвертировать значение невозможно.	Конвертирует вещественное число в строковое значение.
AsInteger	Возвращает значение атрибута, конвертированное в целое число. Выдает исключение, если конвертировать значение невозможно.	Конвертирует целое число в строковое значение.
AsString	Значение атрибута.	Значение атрибута.
Value	Значение атрибута в виде строки.	Конвертирует присваиваемое значение в строку. При невозможности конвертации генерирует исключение.

* Значение строки зависит от текущих региональных настроек в системе.

4.4.2 Целые числа (ftInteger и ftAutoInc)

Атрибут, тип данных которого представляет собой целочисленные значения (**Int64**), позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения-записи своего значения:

Использование свойств атрибута с целочисленным значением.

Свойство	Чтение	Запись
AsBoolean	true если AsInteger <> 0; false если AsInteger = 0; выдает исключение если значение пустое.	При записи true присваивает атрибуту 1. При записи false присваивает 0.
AsDateTime	Операция недопустима.	Операция недопустима.
AsDouble	Возвращает значение атрибута, конвертированное в вещественное число.	Конвертирует вещественное число в целочисленное значение. Выдает исключение, если конвертировать число невозможно.
AsInteger	Значение атрибута.	Значение атрибута.

AsString	Возвращает значение атрибута, конвертированное в строку.	Конвертирует строку в целочисленное значение. Выдает исключение, если конвертировать строку в целое число невозможно.
Value	Значение атрибута в виде целого числа Int64 .	Конвертирует присваиваемое значение в целое число. При невозможности конвертации генерирует исключение.

4.4.3 Вещественные числа (ftDouble)

Атрибут, тип данных которого представляет собой вещественные значения (**double**), позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения-записи своего значения:

Использование свойств атрибута с вещественным значением.

Свойство	Чтение	Запись
AsBoolean	true если <code>AsInteger <> 0</code> ; false если <code>AsInteger = 0</code> ; выдает исключение если значение пустое.	При записи true присваивает атрибуту 1. При записи false присваивает 0.
AsDateTime	Операция недопустима.	Операция недопустима.
AsDouble	Значение атрибута.	Значение атрибута.
AsInteger	Возвращает значение атрибута, конвертированное в целое число. Выдает исключение, если конвертировать число невозможно.	Конвертирует целое число в вещественное значение.
AsString	Возвращает значение атрибута, конвертированное в строку.	Конвертирует строку в вещественное значение. Выдает исключение, если конвертировать строку в число невозможно.
Value	Значение атрибута в виде вещественного числа double .	Конвертирует присваиваемое значение в вещественное число. При невозможности конвертации генерирует исключение.

4.4.4 Дата и время (ftDateTime)

Атрибут, тип данных которого представляет собой дату и время (**DateTime**), позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения-записи своего значения:

Использование свойств атрибута даты и времени.

Свойство	Чтение	Запись
AsBoolean	Операция недопустима.	Операция недопустима.
AsDateTime	Значение атрибута в локальном времени.	Значение атрибута в локальном времени.
AsDouble	Операция недопустима.	Операция недопустима.
AsInteger	Операция недопустима.	Операция недопустима.
AsString*	Возвращает значение атрибута, конвертированное в строку.	Конвертирует строку в дату и время. Выдает исключение, если конвертировать строку невозможно.
Value	Значение атрибута в виде даты и времени DateTime .	Конвертирует присваиваемое значение в дату и время. При невозможности конвертации генерирует исключение.

* Значение строки зависит от текущих региональных настроек в системе.

4.4.5 Тексты произвольной длины (ftMemo)

Атрибут, тип данных которого представляет собой строку, максимальный размер которой ограничен константой **Intermech.Consts.MaxShortBlobSize** (по умолчанию это 131072 символа). Размер может быть уменьшен администратором системы для каждого конкретного атрибута. Читать и модифицировать значения

атрибута можно через функции общего интерфейса **IDBAttribute**, а также посредством специальных интерфейсов **IMemoReader** и **IMemoWriter**, которые позволяют осуществлять пакетное чтение и запись значений атрибута.

Использование свойств текстового атрибута.

Свойство	Чтение	Запись
AsBoolean	Операция недопустима.	Операция недопустима.
AsDateTime	Дата и время последней модификации значения атрибута (в локальном времени).	Операция недопустима.
AsDouble	Операция недопустима.	Операция недопустима.
AsInteger	Идентификатор записи в таблице IMS_MEMOS.	Операция недопустима.
AsString	Первые 450 символов значения атрибута.	Значение атрибута.
Value	Значение атрибута.	Значение атрибута.

4.4.5.1 Интерфейс IMemoReader

Интерфейс пакетного чтения текстовых атрибутов на клиенте.

4.4.5.1.1 Пространство имён

Intermech.Interfaces

4.4.5.1.2 Методы интерфейса

CloseMemo

Закрывает текстовое поле и высвобождает занятые ресурсы.

Синтаксис:

```
void CloseMemo();
```

OpenMemo

Открывает текстовое поле и возвращает его размер в символах.

Синтаксис:

```
int OpenMemo(int dataBlockSize);
```

Параметры:

`dataBlockSize`

Размер блоков в символах, которые он будет отдавать на клиента при чтении. Если значение параметра **dataBlockSize** равно **0**, то размер блока устанавливается равным размеру реально хранимых данных.

ReadDataBlock

Читает следующий блок данных размера **dataBlockSize**. Если значение **dataBlockSize** равно **0**, то размер блока задается в параметре функции **OpenMemo**, которая обязательно должна быть вызвана перед **ReadDataBlock**. Если длина возвращаемого блока меньше **dataBlockSize** или равна **0**, то данные закончились и текстовое поле закрыто.

Синтаксис:

```
char[] ReadDataBlock(int dataBlockSize);
```

```
char[] ReadDataBlock();
```

Параметры:

`dataBlockSize`

Размер читаемого блока данных в символах.

4.4.5.2 Интерфейс IMemoWriter

Интерфейс пакетной записи текстовых полей.

4.4.5.2.1 Пространство имён

Intermech.Interfaces

4.4.5.2.2 Методы интерфейса

CancelWrite

Закрывает текстовое поле и отменяет запись данных.

Синтаксис:

```
void CancelWrite();
```

OpenMemo

Открывает текстовое поле для записи. Функция возвращает значение **true**, если поле открыто и готово к записи данных.

Синтаксис:

```
bool OpenMemo(int memoSize);
```

Параметры:

memoSize

Количество записываемых символов.

Результат:

true, если поле открыто и готово к записи данных.

WriteDataBlock

Дописывает блок символов в текстовое поле.

Синтаксис:

```
bool WriteDataBlock(char[] data);
```

Параметры:

data

Блок записываемых символов. Если размер данных превышает размер, указанный в методе **OpenMemo**, то генерируется исключение.

Результат:

Если возвращает **true**, то запись текста не завершена и ожидается продолжение данных. Если размер записанных данных сравнялся с заявленным размером текстового поля, то поле для записи закрывается, а функция возвращает **false**.

4.4.6 Глобальные идентификаторы (ftGuid)

Атрибут, тип данных которого представляет собой глобальный идентификатор (**Guid**), позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения-записи своего значения:

Использование свойств атрибута типа **Глобальный идентификатор**.

Свойство	Чтение	Запись
AsBoolean	Операция недопустима.	Операция недопустима.
AsDateTime	Операция недопустима.	Операция недопустима.
AsDouble	Операция недопустима.	Операция недопустима.
AsInteger	Операция недопустима.	Операция недопустима.
AsString	Возвращает значение атрибута, конвертированное в строку.	Конвертирует строку в глобальный идентификатор. Выдает исключение, если конвертировать строку невозможно.
Value	Возвращает строковое представление значения атрибута.	Конвертирует присваиваемое значение в глобальный идентификатор. Выдает исключение, если конвертировать строку невозможно.

Также атрибуты типа **ftGuid** поддерживают интерфейс **IDBGuidAttribute**, который позволяет читать и изменять значение атрибута, используя тип **System.Guid**.

4.4.7 Ссылки на версию объектов (ftObjectLink)

Атрибут, тип данных которого представляет собой ссылку на версию информационного объекта, позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения и записи своего значения:

Использование свойств атрибута типа **Ссылка на версию объекта**.

Свойство	Чтение	Запись
AsBoolean	Операция недопустима.	Операция недопустима.
AsDateTime	Операция недопустима.	Операция недопустима.
AsDouble	Операция недопустима.	Операция недопустима.

AsInteger	Идентификатор версии объекта, на которую ссылается атрибут.	Идентификатор версии объекта, на которую ссылается атрибут.
AsString	Заголовок версии объекта, на которую ссылается атрибут. Заполняется ядром системы автоматически.	Операция недопустима.
Value	Идентификатор версии объекта, на которую ссылается атрибут.	Допускается присвоение локального или глобального идентификатора версии объекта, а также интерфейса IDBObject .

Внимание! Ссылочные атрибуты всегда содержат ссылки на архивные копии объекта, поэтому нельзя ссылаться на заготовки объектов. При записи в атрибут ссылки на рабочую копию объекта система автоматически преобразует ее в ссылку на архивную копию этой же версии объекта.

4.4.7.1 Интерфейс **IDBObjectLinkAttribute**

Интерфейс, поддерживаемый атрибутами типа **Ссылка на версию объекта**.

4.4.7.1.1 Пространство имён

Intermech.Interfaces

4.4.7.1.2 Свойства интерфейса

IDBObject

Объект, на который ссылается текущее значение атрибута. Если этот объект взят данным пользователем на изменение, то возвращается рабочая копия объекта. Если атрибут содержит пустое значение, то свойство вернёт **null**.

Синтаксис:

```
IDBObject IDBObject { get; set; }
```

IDBObjectGUID

Глобальный идентификатор версии объекта, на которую ссылается текущее значение атрибута. Если атрибут содержит пустое значение, то свойство вернёт значение **Guid.Empty**.

Синтаксис:

```
Guid IDBObjectGUID { get; set; }
```

4.4.8 Пароли (ftPassword)

Атрибут, предназначенный для хранения паролей, позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения и записи своего значения:

Использование свойств атрибута типа **Пароль**.

Свойство	Чтение	Запись
AsBoolean	Операция недопустима.	Операция недопустима.
AsDateTime	Операция недопустима.	Операция недопустима.
AsDouble	Операция недопустима.	Операция недопустима.
AsInteger	Операция недопустима.	Операция недопустима.
AsString	Возвращает хэш-код сохраненного пароля. Первым идет символ, обозначающий метод хэширования пароля ('0' – пароль хранится в открытом виде).	Строка пароля. Метод хэширования устанавливается администратором системы для всех атрибутов типа Пароль .
Value	Возвращает строку с хэш-кодом пароля.	Строка пароля.

4.4.8.1 Интерфейс **IDBEncryptedAttribute**

Интерфейс, поддерживаемый атрибутами типа **Пароль**.

4.4.8.1.1 Пространство имён

Intermech.Interfaces

4.4.8.1.2 Методы интерфейса

ValidateCurrent

Метод проверяет, содержится ли в данный момент в атрибуте хэш-код строки **nowValue**.

Синтаксис:

```
bool ValidateCurrent(string nowValue);
```

Параметры:

nowValue

Проверяемая строка.

Результат:

Метод вернет **true**, если в атрибуте хранится хэш-код указанной строки.

ValidateNew

Метод генерирует исключение в случае, если проверяемый пароль не соответствует текущей политике безопасности системы.

Синтаксис:

```
void ValidateNew(string newValue);
```

Параметры:

newValue

Строка с паролем.

SetPassword

Устанавливает новый пароль.

Синтаксис:

```
void SetPassword(PswPackage psw);
```

Параметры:

psw

Хэши пароля.

4.4.9 Вещественные числа, выраженные в единицах измерения (ftMeasured)

Атрибут, предназначенный для хранения чисел, выраженных в единицах измерения. Позволяет использовать следующие свойства интерфейса **IDBAttribute** для чтения и записи своего значения:

Использование свойств атрибута типа **ftMeasured**.

Свойство	Чтение	Запись
AsBoolean	Операция недопустима.	Операция недопустима.
AsDateTime	Операция недопустима.	Операция недопустима.
AsDouble	Возвращает вещественное значение атрибута, приведённое к базовой единице измерения.	Операция недопустима.
AsInteger	Конвертирует вещественное значение атрибута в целое число, приведённое к базовой единице измерения.	Операция недопустима.
AsString	Возвращает строковое представление значения атрибута в том виде, в котором его ввел пользователь.	Преобразует строку в число, выраженное в единицах измерения. Выдает исключение, если конвертировать строку невозможно.
Value	Возвращает значение атрибута в виде класса MeasuredValue , в котором значение приведено к базовой единице измерения, а поле Caption хранит значение в той физической величине, в которой его ввел в атрибут пользователь. Если атрибут содержит пустое значение, то возвращает DBNull.Value .	Может принимать как строковое представление числа с указанной единицей измерения, так и класс MeasuredValue .

4.4.9.1 Класс *MeasuredValue*

Класс с описанием значения величины, выраженной в единицах измерения.

4.4.9.1.1 Пространство имён

Intermech.Interfaces

4.4.9.1.2 Свойства класса

Caption

Строковое представление значения и его единицы измерения (например, 1 кг).

Синтаксис:

```
public string Caption;
```

MeasureID

Идентификатор единицы измерения. Может не совпадать с единицей измерения, которая указана в свойстве **Caption**, т.к. атрибуты типа **ftMeasured** хранят в базе значения, приведённые к базовым единицам измерения, а на экран выводят строки в тех единицах, которые указал при вводе пользователь. Например, если базовая единица измерения массы это килограммы, а пользователь ввёл значение **4 т**, то после чтения этого значения из атрибута свойство **MeasureID** будет иметь идентификатор килограммов, свойство **Value** будет содержать значение 4000, а в **Caption** будет записана строка **4 т**.

Синтаксис:

```
public Int64 MeasureID;
```

Value

Значение величины в данной единице измерения.

Синтаксис:

```
public double Value.
```

4.4.9.1.3 Методы класса

Clone

Возвращает полную копию данного класса **MeasuredValue**.

Синтаксис:

```
public object Clone();
```

4.4.9.2 Интерфейс **IDBMeasureAttribute**

Интерфейс атрибута, хранящего числовое значение вместе с его единицей измерения. В строковой составляющей атрибута хранится строковое представление хранящегося значения в той единице измерения, в которой ввел его пользователь. В вещественной составляющей - значение, приведенное к базовой величине. В целочисленной хранится идентификатор базовой единицы измерения, в которой записано значение атрибута. Базовой считается единица измерения физической величины с коэффициентом приведения = 1.

4.4.9.2.1 Пространство имён

Intermech.Interfaces

4.4.9.2.2 Свойства интерфейса

MeasureID

Идентификатор объекта, описывающего единицу измерения, в которой хранится значение атрибута. При присвоении идентификатора другой единицы измерения переопределяет строковое представление значения атрибута, если оно совместимо с текущей единицей измерения. Совместимыми считаются единицы измерения одной и той же физической величины. Например, килограмм совместим с граммом, но не совместим метром.

Синтаксис:

```
Int64 MeasureID{ get; set; }
```

MeasureName

Полное наименование базовой единицы измерения (например, “килограмм”).

Синтаксис:

```
string MeasureName{ get; }
```

MeasureShortName

Короткое имя базовой единиц измерения (например, “кг”).

Синтаксис:

```
string MeasureShortName{ get; }
```

Value

Значение атрибута.

Синтаксис:

```
MeasuredValue Value { get; set; }
```

4.4.9.2.3 Методы интерфейса

IsCompatible

Проверяет единицу измерения на совместимость с данным атрибутом, т.е. возможно ли число, выраженное в данной единице измерения, записать в данный атрибут.

Синтаксис:

```
bool IsCompatible(Int64 aMeasureID);
```

Параметры:

aMeasureID

Идентификатор объекта-единицы измерения, который проверяется на совместимость с данным атрибутом.

4.4.9.3 Класс MeasureDescriptor

Описатель единицы измерения. Получается с помощью класса **MetadataHelper**, который описан ниже.

4.4.9.3.1 Пространство имён

Intermech.Interfaces

4.4.9.3.2 Свойства класса

IsDefault

Свойство определяет, используется ли данная единица измерения по умолчанию.

Синтаксис:

```
public bool IsDefault;
```

K

Коэффициент приведения к базовой величине. Это число, на которое нужно умножить базовую единицу измерения данной физической величины, чтобы получить данную единицу измерения. Для базовой единицы измерения коэффициент равен 1.

Синтаксис:

```
public double K;
```

LongName

Наименование единицы измерения (например, “килограмм”).

Синтаксис:

```
public string LongName;
```

MeasureID

Идентификатор объекта-описателя данной единицы измерения.

Синтаксис:

```
public Int64 MeasureID;
```

OperationsList

Список математических операций над единицами измерения, в результате которых получается данная единица измерения. В каждой строке можно использовать только одну операцию и две единицы измерения, которые записываются в виде кратких наименований. Например, для получения кубического метра можно записать формулу в виде **м2*м**. Эта информация используется в дальнейшем системой для определения единиц измерения, которые получаются в результате вычисления значений атрибутов, выраженных в единицах измерения.

Синтаксис:

```
public string[] OperationsList;
```

PhysicalQuantityID

Идентификатор физической величины, которую измеряют с помощью данной единицы измерения.

Синтаксис:

```
public Int64 PhysicalQuantityID;
```

ShortNameIndex

Нормализованное краткое наименование единицы измерения. Вычисляется системой с использованием общих правил построения нормализованных индексов.

Синтаксис:

```
public string ShortNameIndex;
```

ShortName

Краткое наименование единицы измерения (например, кг).

Синтаксис:

```
public string ShortName;
```

4.4.9.4 Класс *MeasureHelper*

Статический класс, который занимается обработкой единиц измерения на сервере и клиенте. Содержит список единиц измерения, зарегистрированных в базе данных IPS Development Kit.

4.4.9.4.1 Пространство имён

Intermech.Interfaces

4.4.9.4.2 Методы класса

Add

Возвращает результат сложения операндов, выраженный в базовой единице измерения результирующей физической величины. При несовместимости операндов генерирует исключение.

Синтаксис:

```
public static MeasuredValue Add (MeasuredValue operand1, MeasuredValue operand2);
```

Параметры:

operand1, operand2

Складываемые значения.

Compare

Возвращает результат сравнения двух значений, выраженных в единицах измерения.

Синтаксис:

```
public static CompareResult Compare (MeasuredValue val1, MeasuredValue val2);
```

Параметры:

val1, val2

Сравниваемые значения.

Результат:

Возможные результаты сравнения:

- CompareResult.Equal – значения равны;
- CompareResult.Less – val1 меньше val2;
- CompareResult.More – val1 больше val2;
- CompareResult.NotCompatible – val1 и val2 несовместимы.

ConvertToMeasuredValue

Приводит величину строкового типа **mValue** к структуре **MeasuredValue** (см. п. [4.4.9.1](#)). Единица измерения должна быть представлена в строке кратким или полным наименованием.

Синтаксис:

```
public static MeasuredValue ConvertToMeasuredValue(string mValue);
```

Параметры:

mValue

Строковое значение величины.

ConvertToString

Приводит вещественное значение **val** к строковому представлению этого значения в единицах измерения **measureID** в виде “Значение Краткое_наименование_единицы_измерения” (например, 5 кг). Если заданная единица измерения не найдена, возвращает пустую строку.

Синтаксис:

```
public static string ConvertToString(double val, Int64 measureID, bool convertFromBase);
```

Параметры:

val

Вещественное значение величины.

measureID

Идентификатор единицы измерения.

convertFromBase

Если значение параметра равно **true**, то метод конвертирует значение из базовой величины в величину с единицей измерения **measureID**.

Divide

Возвращает результат деления операндов, выраженный в базовой единице измерения результирующей физической величины.

Синтаксис:

```
public static MeasuredValue Divide(MeasuredValue operand1, MeasuredValue operand2);
```

Параметры:

operand1, operand2

Делимое и делитель.

FindBaseValue

Возвращает описатель базовой единицы измерения для описателя **mDescriptor**.

Синтаксис:

```
public static MeasureDescriptor FindBaseValue(MeasureDescriptor mDescriptor);
```

FindDescriptor

Находит описатель единицы измерения по ее значению, идентификатору, краткому наименованию. Если описатель не найден, то возвращаемое значение свойства **MeasureDescriptor.Empty** равно **true**.

Синтаксис:

```
public static MeasureDescriptor FindDescriptor(MeasuredValue mValue);
```

```
public static MeasureDescriptor FindDescriptor(Int64 measureID);
```

```
public static MeasureDescriptor FindDescriptor(string shortName);
```

Параметры:

mValue

Значение с единицей измерения.

measureID

Идентификатор единицы измерения.

shortName

Краткое наименование единицы измерения.

Multiply

Возвращает результат умножения операндов, выраженный в базовой единице измерения результирующей физической величины.

Синтаксис:

```
public static MeasuredValue Multiply(MeasuredValue operand1, MeasuredValue operand2);
```

Параметры:

operand1, operand2

Перемножаемые величины.

ReplaceDescriptor

Заменяет описатель единицы измерения mDescriptor (используется, если изменились свойства описателя). Возвращает значение **true**, если замена была осуществлена.

Синтаксис:

```
public static bool ReplaceDescriptor(MeasureDescriptor mDescriptor);
```

Параметры:

mDescriptor

Описатель единицы измерения.

Subtract

Возвращает результат вычитания операндов, выраженный в базовой единице измерения результирующей физической величины.

Синтаксис:

```
public static MeasuredValue Subtract(MeasuredValue operand1, MeasuredValue operand2);
```

Параметры:

operand1, operand2

Величины с единицами измерения.

4.4.10 Файлы и двоичные данные (ftFile, ftBlob, ftShortBlob)

Атрибуты, предназначенные для хранения файлов и двоичных данных. Атрибуты типа **ftFile** и **ftBlob** хранятся в файловых шкафах IPS Development Kit и ограничены в размере значения только в силу ограничений, которые накладывает сам файловый шкаф (например, СУБД, в которой этот файловый шкаф

расположен). Значения атрибута типа **ftShortBlob** хранится в системной таблице **IMS_BLOBS** и имеет ограничение на размер в 128Kb (при этом администратор для каждого атрибута может уменьшить допустимый размер хранимых в атрибуте данных). Однако атрибуты типа **ftShortBlob** обеспечивают более быстрый доступ к данным, т.к. не требуют промежуточного хранения данных на диске сервера приложений. Для чтения заголовков двоичных атрибутов можно использовать следующие свойства интерфейса **IDBAttribute**:

Использование свойств атрибутов, предназначенных для хранения файлов и двоичных данных.

Свойство	Чтение	Запись
AsBoolean	Операция недопустима.	Операция недопустима.
AsDateTime	Возвращают локальную дату и время последней модификации значения атрибута.	Операция недопустима.
AsDouble	Возвращает идентификатор файлового шкафа, в котором хранится текущее значение атрибута (для атрибутов типа ftFile и ftBlob).	Операция недопустима.
AsInteger	Возвращает размер текущего файла или двоичных данных (в байтах).	Операция недопустима.
AsString	Возвращает имя файла (для атрибутов типа ftFile) или комментарии к двоичным данным (для ftBlob и ftShortBlob).	Операция недопустима.
Value	Возвращает имя файла (для атрибутов типа ftFile) или комментарии к двоичным данным (для ftBlob и ftShortBlob).	Операция недопустима.

Для записи значений в двоичные и файловые атрибуты предназначен интерфейс **IBlobWriter**, который описан ниже.

4.4.10.1 Перечислитель BlobAttributeStates

Перечисления состояний, которые может принимать атрибут с двоичными данными. (закрыт, открыт для чтения данных, открыт для записи данных).

4.4.10.1.1 Пространство имен

Intermech.Interfaces

4.4.10.1.2 Значения перечислителя:

Closed = 0x00000000
BLOB-поле закрыто.
OpenedForRead = 0x00000001
BLOB-поле открыто для чтения.
OpenedForWrite = 0x00000002
BLOB-поле открыто для записи.

4.4.10.2 Перечислитель ArcMethods

Метод упаковки информации в BLOB-поле.

4.4.10.2.1 Пространство имен

Intermech

4.4.10.2.2 Значения перечислителя:

NotPacked = 0x00000000
Без сжатия.
ZLibPacked = 0x00000001
Информация упакована с помощью библиотеки ZLib.

4.4.10.3 Структура BlobInformation

Структура с информацией о файле (BLOB-поле).

4.4.10.3.1 Пространство имен

Intermech.Interfaces

4.4.10.3.2 Поля структуры:

Intermech.ArcMethods ArcMethod

Метод запаковки.

Int64 BlobID

Идентификатор записи о файле (BLOB-поля).

string FileName

Имя файла (для BLOB-поля пусто).

string Note

Комментарии (только для тех, что хранятся в файловом хранилище).

Int64 PackedFileSize

Упакованный размер файла (BLOB-поля).

Int64 RealFileSize

Реальный размер файла (BLOB-поля).

FileType

Тип файла

Синтаксис:

FileTypes FileType

Author

Автор файла (ObjectID)

Синтаксис:

Int64 Author

4.4.10.3.3 Свойства структуры:

System.DateTime ModifyDate { get; set; }

Дата модификации файла (BLOB-поля) в локальном времени с округлением до секунд (чтобы потом правильно работало сравнение).

4.4.10.3.4 Методы структуры:

BlobInformation (конструктор)

Создать экземпляр структуры.

Синтаксис:

```
void BlobInformation (Int64 realFileSize; Int64 packedFileSize;
```

```
System.DateTime modifyDate;
```

```
string fileName;
```

```
Intermech.ArcMethods arcMethod;
```

```
string note)
```

Параметры:

realFileSize

Реальный размер файла (BLOB-поля).

packedFileSize

Упакованный размер файла (BLOB-поля).

modifyDate

Дата модификации файла (BLOB-поля) в локальном времени с округлением до секунд.

fileName

Имя файла (для BLOB-поля пусто).

arcMethod

Метод запаковки.

note

Комментарии (только для тех, что хранятся в файловом хранилище).

Clone

Создать копию экземпляра структуры.

Синтаксис:

Intermech.Interfaces.BlobInformation Clone ()

Возвращаемое значение:

Копия экземпляра структуры.

EmptyBlobInformation

Метод создает пустую структуру с информацией о BLOB-поле

Синтаксис:

```
public static BlobInformation EmptyBlobInformation()
```

Результат:

Пустая структура с информацией о BLOB-поле.

4.4.10.3.5 Статические методы структуры:

EmptyBlobInformation

Создать пустую структуру с информацией о BLOB-поле.

Синтаксис:

```
static Intermech.Interfaces.BlobInformation EmptyBlobInformation ()
```

Возвращаемое значение:

Пустая структура с информацией о BLOB-поле.

4.4.10.4 Интерфейс IBlobReader

Интерфейс чтения BLOB-полей и файлов на клиентской стороне IPS Development Kit.

4.4.10.4.1 Пространство имен

Intermech.Interfaces

4.4.10.4.2 Методы интерфейса:

CloseBlob

Закрывает BLOB-поле и освобождает занятые ресурсы.

Синтаксис:

```
void CloseBlob ()
```

OpenBlob

Открывает BLOB-поле (файл) и возвращает блок с информацией о нем. dataBlockSize – размер блоков, которые он будет отдавать при чтении. Если dataBlockSize меньше 0, то читается только информация, а блок-атрибут остается закрытым для чтения. Если dataBlockSize=0, то размер блока равен размеру реально хранимых в блоке данных.

Синтаксис:

```
Intermech.Interfaces.BlobInformation OpenBlob (Int32 dataBlockSize)
```

Параметры:

dataBlockSize

Размер блоков, которые BLOB-поле будет отдавать при чтении.

Возвращаемое значение:

Информация о BLOB-поле.

ReadDataBlock

Читает следующий блок данных.

Синтаксис:

```
byte[] ReadDataBlock ()
```

Возвращаемое значение:

Загруженные данные.

ReadDataBlock

Читает следующий блок данных размером dataBlockSize. Если dataBlockSize==0, то размер блока берется заданный в OpenBlob. Если возвращенный блок имеет длину 0 (или меньше dataBlockSize), то данные закончились и блок закрыт.

Синтаксис:

```
byte[] ReadDataBlock (Int32 dataBlockSize)
```

Параметры:

dataBlockSize

Размер блока данных для чтения.

Возвращаемое значение:

Загруженные данные.

4.4.10.5 Интерфейс *IBlobWriter*

Интерфейс записи BLOB-полей и файлов.

4.4.10.5.1 Пространство имен

Intermech.Interfaces

4.4.10.5.2 Методы интерфейса:

CancelWrite

Закрывает BLOB-поле и отменяет запись данных.

Синтаксис:

```
void CancelWrite ()
```

OpenBlob

Открывает BLOB-поле для записи и инициализирует его информацией из блока blobInfo. Если onlyInfo=true, то модифицируется только сопутствующая информация (filename и Note). Ф-ция возвращает true, если BLOB-поле ждет данные для записи, иначе BLOB-поле сразу закрывается (если потребовали записать 0 байт).

Синтаксис:

```
bool OpenBlob (Intermech.Interfaces.BlobInformation blobInfo; bool onlyInfo)
```

Параметры:

blobInfo

Информация о BLOB-поле.

onlyInfo

Если равно true, то модифицируется только сопутствующая информация.

Возвращаемое значение:

true, если BLOB-поле ждет данные для записи, иначе BLOB-поле сразу закрывается.

WriteDataBlock

Дописывает блок данных в BLOB-поле. Если возвращает true, то запись BLOB-поля не завершена и ожидается продолжение данных, иначе записывает данные в BLOB-поле. Если data будет содержать данные более размера BLOB-поля, то будет выдано исключение.

Синтаксис:

```
bool WriteDataBlock (byte[] data)
```

Параметры:

data

Записываемые данные.

Возвращаемое значение:

Если возвращает **true**, то запись BLOB-поля не завершена и ожидается продолжение данных.

4.4.11 Логический тип данных (ftBoolean)

Атрибут, тип данных которого является логическим (**bool**), позволяет использовать свойства интерфейса **IDBAttribute** для чтения-записи своего значения различными способами.

Использование свойств логического атрибута.

Свойство	Чтение	Запись
AsBoolean	Значение атрибута.	Значение атрибута.
AsDateTime	Операция недопустима.	Операция недопустима.
AsDouble	Возвращает 1 если атрибут содержит true , и 0 – если false .	Записывает в атрибут true , если свойству присваивают ненулевое значение, иначе записывает false .
AsInteger	Возвращает 1 если атрибут содержит true , и 0 – если false .	Записывает в атрибут true , если свойству присваивают ненулевое значение, иначе записывает false .
AsString*	Возвращает строку Да если атрибут содержит true , и Нет – если атрибут содержит false .	Конвертирует строку в логическое выражение. В качестве значения true принимаются строки

		Истина, Да, True. В качестве значения false принимаются Ложь, Нет, False.
Value	Возвращает значение атрибута типа bool (или DBNull.Value в случае если атрибут содержит пустое значение).	Конвертирует присваиваемый объект в логическое значение атрибута. При невозможности конвертации выдает исключение.

* Регистр букв не имеет значения.

5 Информационные объекты

5.1 Базовые перечислители, структуры, классы, интерфейсы

5.1.1 Перечислитель AttributeSourceTypes

Перечислитель описывает принадлежность атрибутов объектам, связям, т.п. Для получения принадлежности обязательных атрибутов используйте метод **AttributeSourceTypes.GetAttributeSourceType**. Все необязательные атрибуты могут принадлежать объектам или связям.

5.1.1.1 Пространство имён

Intermech.Kernel.Search

5.1.1.2 Значения перечислителя

Auto = 0
Источник не указан.
Object = 1
Атрибут объекта.
Relation = 2
Атрибут связи.
Events = 3
Атрибут события.
History = 4
Атрибут истории значений.
FileStorage = 5
Атрибут файлового шкафа.
Other = 6
Прочие атрибуты.

5.1.2 Перечислитель ColumnContents

Перечислитель предназначен для указания той части значений сложных атрибутов, с которой программе требуется работать в данный момент.

5.1.2.1 Пространство имён

Intermech.Kernel.Search

5.1.2.2 Значения перечислителя

Text = 0

Текстовое представление атрибута, предназначенное для показа значений пользователям. Например, для ссылочных атрибутов возвращается строковая часть значения, которая содержит заголовки объектов. Для простых атрибутов используются сами значения атрибутов – строки, целые и вещественные числа, даты. Данное значение перечислителя применяется системой по умолчанию.

ID = 1

Идентификационная часть атрибута (например, идентификатор версии объекта, на которую ссылается атрибут типа ftObjectLink). Может использоваться только для атрибутов типа ftObjectLink, ftBlob, ftShortBlob, ftFile, ftMeasured.

Date = 2

Информация о дате модификации атрибута (только для типов ftBlob, ftShortBlob и ftFile).

Value = 3

Для ftBlob и ftFile содержит идентификатор файлового шкафа, для ftMeasured - значение атрибута, приведенное к базовой единице измерения. Для остальных типов значение не применимо!

String = 4

Явное указание на использование строковой составляющей атрибута.

5.1.3 Перечислитель ColumnNameMapping

Перечислитель задаёт способ именования колонки атрибута в DataTable, возвращаемый методом IDBRecords.Select.

5.1.3.1 Пространство имён

Intermech.Kernel.Search

5.1.3.2 Значения перечислителя

Default = 0

Заголовок колонки будет зависеть от того, какая часть значения атрибута будет содержаться в колонке. Для ColumnContents.Text заголовком будет наименование атрибута. Для ColumnContents.ID к имени атрибута добавляется слово (**идентификатор**), для ColumnContents.Date – фраза (**дата модификации**).

ID = 1

Целочисленный идентификатор атрибута.

Guid = 1

Глобальный идентификатор атрибута.

Alias = 2

Псевдоним атрибута (если оно есть).

ShortName = 3

Краткое наименование атрибута (если оно есть).

Name = 4

Наименование атрибута.

FieldName = 5

Имя колонки в базе данных (только для обязательных атрибутов!).

Index = 6

Порядковый номер колонки в DataTable. Рекомендуется использовать данный способ именования колонок если требуется гарантировать уникальность наименований. В остальных случаях система не гарантирует уникальности, т.к. один и тот же атрибут может участвовать в запросе несколько раз.

5.1.4 Перечислитель LogicalOperators

Перечислитель позволяет указать логический оператор (например, для объединения условий поиска объектов).

5.1.4.1 Пространство имён

Intermech.Kernel.Search

5.1.4.2 Значения перечислителя

NONE = 0

Нет логического оператора.

OR = 1

Оператор **ИЛИ**.

AND = 2

Оператор **И**.

NOT = 3

Оператор **НЕ**.

5.1.5 Перечислитель ObjectFiltrationState

Перечислитель описывает статус последней фильтрации текущего объекта по правилу подбора версий.

5.1.5.1 Пространство имён

Intermech.Interfaces

5.1.5.2 Значения перечислителя

fsNotRequired = 0

Фильтрация по правилам подбора версий не требуется. Выбрано правило "Все версии объектов".

fsCompositeVersionNotFound = 1

Не найдена версия объекта в переданных данных при конкретизации версии в составе.

fsFiltrationStopped = 2

Для данной версии фильтрация остановлена из-за того, что версия не удовлетворяет основным критериям подбора.

fsVersionNotFound = 3

Не найдена версия объекта в переданных данных.

fsInvalidRule = 4

Не удалось получить правило подбора версий для фильтрации.

fsVariance = 5

Не существует версий соответствующих основному критерию подбора. Версия отобрана по дополнительному критерию подбора.

fsNonVersionable = 6

Объекту не требуется фильтрация по правилам подбора версий. Объект неверсионный.

fsCorrespondingSingle = 7

Объект полностью соответствует правилу подбора версий. Причём он единственный из версий, прошедший фильтрацию по основным критериям подбора.

fsCorresponding = 8

Существует несколько версий объекта, соответствующих основному критерию подбора. Версия отобрана по дополнительному критерию подбора.

fsCompositeVersion = 9

Версия подобрана по обязательному атрибуту связи **Идентификатор версии в составе** (подбор выполнен по конкретизации версии на связи).

fsVersionFromMainContext = 10

Версия подобрана по основному контексту редактирования.

fsVersionFromLinkedContext = 11

Версия подобрана по связанному контексту редактирования.

5.1.6 Перечислитель **ObligatoryObjectAttributes**

Перечислитель описывает обязательные атрибуты в системе (у объектов, связей, событий и пр.).

5.1.6.1 *Пространство имён*

Intermech.Kernel.Search

5.1.6.2 *Значения перечислителя*

Имена значений соответствуют именам полей в базе данных, а сами значения являются идентификаторами F_ATTRIBUTE_ID этих атрибутов в общем списке атрибутов системы. Часть атрибутов не хранят значения в базе данных, в вычисляются ядром системы в момент формирования пакета данных (например, F_ELEMENT_STATUSES). Некоторые из значений данного перечислителя соответствуют свойствам интерфейса IDBObject. В таблице ниже приведены соответствия:

Таблица обязательных атрибутов объектов IMS_OBJECTS.

Обязательные атрибуты	Тип поля в СУБД	Примечания	Свойства в IDBObject
F_OBJECT_ID	BIGINT	Идентификатор версии объекта.	ObjectID
F_ID	BIGINT	Идентификатор объекта.	ID
F_LC_STEP	INTEGER	Идентификатор этапа жизненного цикла, на котором находится данная версия объекта.	LCStep
F_VERSION_ID	INTEGER	Порядковый номер версии объекта.	VersionID
F_CHKOUTBY	BIGINT	Идентификатор пользователя, взявшего версию объекта на редактирование.	CheckoutBy
F_OBJECT_VER_TYPE	INTEGER	Статус версии (-1 – заготовка, 0 – версия объекта, 1 – версия находится в процессе импорта из портфеля или web-портала). Значения соответствуют перечислителю Intermech.ObjectRecordKind.	ObjectVerType
F_OBJECT_TYPE	INTEGER	Идентификатор типа объекта.	ObjectType
F_OWNER_ID	BIGINT	Идентификатор версии объекта-владельца.	OwnerID
F_MODIFY_DATE	DATE	Дата последней модификации атрибутов данной версии объекта.	ModifyDate

F_LEVEL_ID	INTEGER	Идентификатор уровня продвижения, на котором находится данная версия объекта.	IDBLifecycleLevel. LevelID
F_OBJ_CREATE	DATETIME	Дата создания версии объекта.	CreateDate
F_PROJECT_ID	BIGINT	Идентификатор проекта, которому принадлежит версия объекта.	ProjectID
F_MODIFICATION_ID	BIGINT	Идентификатор группы изменений, к которой относится версия объекта.	ModificationID
F_BASE_VERSION	BIGINT	Признак базовой версии (1 – базовая версия).	IsBaseVersion
F_SITE_ID	NVARCHAR(2)	Символьные идентификаторы узлов информационной сети, которые являются источником и владельцем данной версии объекта.	SiteID

5.1.7 Перечислитель RelationalOperators

Перечислитель с операторами отношений, предназначенными для создания условий поиска информации в базе данных IPS Development Kit.

5.1.7.1 Пространство имён

Intermech.Kernel.Search

5.1.7.2 Значения перечислителя

None = -1

Нет оператора.

Empty = 0

Атрибут содержит пустое значение.

NotEmpty = 1

Атрибут содержит не пустое значение.

Equal = 2

Равно.

NotEqual = 3

Не равно.

Greater = 4

Больше.

GreaterOrEqual = 5

Больше или равно.

Less = 6

Меньше.

LessOrEqual = 7

Меньше или равно.

Substring = 8

Содержит строку.

StartString = 9

Начинается со строки.

EndString = 10

Заканчивается строкой.

Between = 11

Находится в диапазоне (включая границы).

NotSubstring = 12

Не содержит строку.

NotStartString = 13

Не начинается со строки.

NotEndString = 14

Не заканчивается строкой.

NotBetween = 15
 Не находится в диапазоне.

EntersIn = 16
 Входит в состав указанной версии объекта.

ConsistFrom = 17
 Состоит из указанной версии объекта.

NotEntersInType = 18
 Не входит в объекты указанного типа.

EntersInType = 19
 Входит в объекты указанного типа.

In = 20
 Равно одному из значений.

InSelection = 21
 Включено в выборку (классификатор).

NOP = 22
 Нет операции (заглушка). Используется для пропуска условия.

ConsistFromType = 23
 Состоит из объектов указанного типа.

NotConsistFromType = 24
 Не состоит из объектов указанного типа.

ParentVersionID = 25
 Создана на основе версии.

ObjectTypeFilter = 26
 Искать среди объектов указанного и всех дочерних типов. Условие использует таблицы оптимизации для ускорения поиска объектов и связей.

AttributeExists = 27
 Содержит атрибут.

LastNDays = 28
 Содержит дату за последние N дней.

NotIn = 29
 Не равно ни одному из значений.

NotExistsOrEmpty = 30
 Не содержит атрибут или значение.

ExistsInVersionContext = 31
 Находится в контексте редактирования.

StringTemplate = 32
 Содержит шаблон строки. В шаблоне допускается использовать символы ? (допускается наличие любого символа в данной позиции) и * (строка может начинаться/заканчиваться любыми символами).

5.1.8 Перечислитель SortOrders

Перечислитель с порядком сортировки значений в колонках.

5.1.8.1 Пространство имён

Intermech.Kernel.Search

5.1.8.2 Значения перечислителя

NONE = 0
 Не сортировать.

ASC = 1
 Сортировать по возрастанию.

DESC = 2
 Сортировать по убыванию.

5.1.9 Структура ColumnDescriptor

Структура с информацией о колонке, которую нужно получить методом IDBRecords.Select.

5.1.9.1 Пространство имён

Intermech.Kernel.Search

5.1.9.2 Свойства структуры

AttributeID

Идентификатор атрибута (числовой идентификатор, глобальный идентификатор или полное наименование атрибута).

Синтаксис:

```
public object AttributeID;
```

AttributeSource

Чему принадлежит данный атрибут (объекту, связи или определяется автоматически по типу коллекции объектов или связей).

Синтаксис:

```
public AttributeSourceTypes AttributeSource;
```

ColumnName

Задаёт способ именования колонки атрибута в таблице, возвращаемой методом IDBRecords.Select.

Синтаксис:

```
public ColumnNameMapping ColumnName;
```

Способы именования колонки.

Contents

Вид информации, которую нужно получить у атрибута.

Синтаксис:

```
public ColumnContents Contents;
```

OrderByID

Задаёт приоритет сортировки для данной колонки (если поле Sort <> SortOrders.NONE). Чем меньше значение данного поля, тем выше приоритет сортировки по данному атрибуту.

Синтаксис:

```
public int OrderByID;
```

Sort

Задаёт порядок сортировки данных по этой колонке.

Синтаксис:

```
public SortOrders Sort;
```

5.1.10 Структура ColumnInfo

Структура с информацией об атрибуте, которую нужно передать обрабатывающей программе.

5.1.10.1 Пространство имён

Intermech.Kernel.Search

5.1.10.2 Свойства структуры

AddInfo

Дополнительная информация по атрибуту (зарезервировано для будущего использования).

Синтаксис:

```
public object AddInfo;
```

AttributeID

Идентификатор атрибута (Int32, Guid, строка с псевдонимом или полным наименованием атрибута).

Синтаксис:

```
public object AttributeID;
```

AttributeSource

Принадлежность атрибута объекту, связи, событию, т.п.

Синтаксис:

```
public AttributeSourceTypes AttributeSource;
```

5.1.11 Структура ConditionStructure

Структура, описывающая условие поиска информации в базе данных.

5.1.11.1 Пространство имён

Intermech.Kernel.Search

5.1.11.2 Свойства структуры

Attribute

Идентификатор атрибута (числовой идентификатор, глобальный идентификатор или полное наименование атрибута).

Синтаксис:

```
public object Attribute;
```

AttributeSource

Указывает на источник атрибута.

Синтаксис:

```
public AttributeSourceTypes AttributeSource;
```

CaseSensitive

Указывает на необходимость поиска строки с учетом регистра букв. Для нестроковых атрибутов значение поля игнорируется.

Синтаксис:

```
public bool CaseSensitive;
```

Content

Позволяет указать ту часть значения комплексных атрибутов, по которой будет производится поиск информации. По умолчанию используется значение ColumnContents.Text.

Синтаксис:

```
public ColumnContents Content;
```

GroupID

Управляет группировкой условий в массивах условий. Положительное значение поля указывает на количество открывающих скобок, которое будет подставлено в формируемое SQL-выражение перед данным условием поиска. Отрицательное значение указывает количество закрывающих скобок после данного условия. По умолчанию поле содержит 0.

Синтаксис:

```
public int GroupID;
```

LogicalOperator

Логический оператор, которым это условие объединяется со следующим по списку условий.

Синтаксис:

```
public LogicalOperators LogicalOperator;
```

RelationalOperator

Оператор отношений.

Синтаксис:

```
public RelationalOperators RelationalOperator;
```

SQL

SQL-оператор для поиска значений (зарезервирован для будущего использования).

Синтаксис:

```
public string SQL;
```

TypeID

Используется для идентификации типа связей в условиях RelationalOperators.EntersInType, RelationalOperators.EntersIn, RelationalOperators.ConsistFromType, RelationalOperators.ConsistFrom и пр.

Синтаксис:

```
public object TypeID;
```

Value

Искомое значение. Тип значения должен соответствовать типу данных атрибута, по которому идет поиск. Для операторов RelationalOperators.In и RelationalOperators.NotIn, RelationalOperators.EntersInType, RelationalOperators.NotEntersInType может содержать массивы значений соответствующего типа.

Синтаксис:

```
public object Value;
```

Value2

Второе искомое значение (например, для операторов RelationalOperators.Between, RelationalOperators.NotBetween).

Синтаксис:

```
public object Value2;
```

5.1.11.3 Методы структуры

Join

Статический метод позволяет выполнить объединение нового условия или массива новых условий с массивом существующих условий.

Синтаксис:

```
static ConditionStructure[] Join (ConditionStructure joinedCondition, ConditionStructure[]  
existingConditions);  
static ConditionStructure[] Join (ConditionStructure[] joinedConditions, ConditionStructure[]  
existingConditions);
```

Параметры:

joinedCondition

Новое условие, которое будет добавлено в конец массива существующих условий.

joinedConditions

Массив новых условий, которые будут добавлены в конец массива существующих условий.

existingConditions

Массив существующих условий, к которому добавляются новые условия. Допускается значение **null**.

Результат:

Новый массив, в котором находятся объединённые условия.

5.1.12 Структура DBRecordSetParams

Структура свойств, описывающих выборку записей из базы данных с помощью методов IDBRecords.Select. Содержит информацию о запрашиваемых атрибутах, условиях поиска, сортировки, именования колонок и пр.

5.1.12.1 Пространство имён

Intermech.Kernel.Search

5.1.12.2 Свойства структуры

ColumnNames

Способы именования колонок атрибутов, указанных в массиве Columns. Размер массива должен совпадать с массивом Columns. Если ColumnNames = null, то используется способ именования ColumnNamesMapping.Default.

Синтаксис:

```
public ColumnNameMapping[] ColumnNames;
```

Columns

Массив с идентификаторами атрибутов (локальные идентификаторы, глобальные идентификаторы или полные наименования атрибутов), которые должны быть включены в результирующую таблицу данных.

Синтаксис:

```
public object[] Columns;
```

ColumnsInfo

Дополнительная информация по колонкам. Массив заполняется только при необходимости. Размер массива должен совпадать с массивом Columns.

Синтаксис:

```
public ColumnInfo[] ColumnsInfo;
```

Conditions

Условия поиска информации в базе данных.

Синтаксис:

```
public ConditionStructure[] Conditions;
```

Contents

Виды информации, которую требуется получить в колонках. Если массив не инициализирован, используется значение по умолчанию **ColumnContents.Text**. Размер массива должен совпадать с массивом **Columns**.

Синтаксис:

```
public ColumnContents[] Contents;
```

FailIfNotFound

Генерировать ли исключение **AttributeTypeNotFoundException** в случае, если в списках **Columns** и **SortColumns** указан несуществующий в системе атрибут. Если значение равно **false**, то атрибут просто не будет включен в результирующую таблицу. По умолчанию значение равно **true**.

Синтаксис:

```
public bool FailIfNotFound;
```

LastKeyValue

Последнее значение ключевого атрибута (например, идентификатор версии объекта при работе со списком объектов), которое было получено клиентом в предыдущем пакете данных. По умолчанию значение равно **0**. Свойство используется при пакетном чтении данных для идентификации записи, с которой следует читать следующий пакет данных.

Синтаксис:

```
public Int64 LastKeyValue;
```

LastOrderValue

Последнее значение атрибута сортировки, которое было получено клиентом в предыдущем пакете данных (по умолчанию значение равно **null**). Если сортировка ведётся по нескольким атрибутам, то свойству следует присвоить массив значений атрибутов, по которым указана сортировка данных. Свойство используется при пакетном чтении данных, поэтому в остальных случаях должно содержать **null**.

Синтаксис:

```
public object LastOrderValue;
```

Orders

Порядок сортировки значений (если массив не инициализирован, то принимается сортировка по возрастанию – **SortOrders.ASC**).

Синтаксис:

```
public SortOrders[] Orders;
```

RecordCount

Количество записей, которые нужно передать клиенту. Рекомендуется использовать константы класса **QueryConsts**. Значение по умолчанию - **QueryConsts.Default**.

Синтаксис:

```
public int RecordCount;
```

SortColumns

Массив с идентификаторами колонок, по которым нужно отсортировать выборку данных. В качестве идентификаторов атрибутов могут быть локальные или глобальные идентификаторы, а также полные наименования атрибутов. Обязательным условием является наличие этих атрибутов также и в списке атрибутов **Columns**, которые должны быть включены в результирующую таблицу данных.

Синтаксис:

```
public object[] SortColumns;
```

SortContents

Вид информации в атрибуте, по которому идет сортировка. Если массив задан, то его размер должен совпадать с размером массива **SortColumns**.

Внимание! Аналогичные значения для атрибутов должны быть заданы и в массиве **Contents**.

Синтаксис:

```
public ColumnContents[] SortContents;
```

SortSources

Источник атрибутов, по которым идет сортировка. Если массив задан, то его размер должен совпадать с размером массива **SortColumns**.

Синтаксис:

```
public AttributeSourceType[] SortSources;
```

TableName

Имя таблицы, которую вернет функция **Select**. Если данному свойству не заполнять, то система сама присвоит таблице имя в зависимости от категории возвращаемых данных (объекты, связи, события и пр.).

Синтаксис:

```
public string TableName;
```

Tags

Дополнительные данные, которые могут передаваться на серверную сторону со стороны клиентского ядра и клиентских плагинов. Для заполнения данного контейнера можно воспользоваться услугами службы **IClientPluginsService**.

Внимание! В данном контейнере можно передавать запрещающие или разрешающие ключи, которые, например, могут управлять фильтрацией составов и т.п. Если требуется добавить ключ в данный словарь, но это свойство равно **null**, следует создать экземпляр класса **HybridDictionary** и присвоить его данному свойству.

Синтаксис:

```
public HybridDictionary Tags;
```

5.1.12.3 Методы структуры

AddColumnDescriptors

Метод позволяет добавить в условия запроса дополнительные колонки. При этом проверяется наличие таких же колонок в параметрах (исключается повторный запрос одних и тех же колонок с одинаковыми способами именования колонок и источником данных).

Синтаксис:

```
int AddColumnDescriptors (ColumnDescriptor[] AddColumns, List<int> AddedColumnsPos);
```

Параметры:

AddColumns

Коллекция дополнительных колонок.

AddedColumnsPos

Если данный параметр задан, в нём будет возвращён список позиций добавленных колонок.

Результат:

Новый массив, в котором находятся объединённые условия.

IndexOf

Метод позволяет найти в указанной таблице столбец с атрибутом, если известно имя атрибута, его идентификатор или Guid.

Синтаксис:

```
public static Int32 IndexOf (DataTable source, object attribute);
```

Параметры:

source

Таблица, в колонках которой разыскивается атрибут.

attribute

Int32, String, Guid – идентификатор или название разыскиваемого атрибута.

Результат:

-1, если атрибут не найден, иначе порядковый номер колонки атрибута в таблице.

5.1.13 Структура QuickObjectInfo

Структура с краткой информацией о версии объекта. Используется для кэширования данных об объектах на серверной и клиентской сторонах при вызове функции IUserSession.GetObjectInfo.

5.1.13.1 Пространство имен

Intermech.Interfaces

5.1.13.2 Поля структуры:

string Caption

Заголовок объекта.

Int64 ObjectID

Идентификатор версии объекта.

Int32 ObjectTypeID

Тип объекта.

5.1.13.3 Свойства структуры:

`bool Empty { get; }`

Возвращает true, если структура пустая (например, в базе не найден объект ObjectID).

5.1.13.4 Методы структуры:

QuickObjectInfo (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void QuickObjectInfo (Int64 objectID; string caption; Int32 objectTypeID)
```

Параметры:

objectID

Идентификатор версии объекта.

caption

Заголовок объекта.

objectTypeID

Тип объекта.

5.1.14 Класс ObligatoryObjectAttributesHelper

Вспомогательный класс для работы с обязательными атрибутами.

5.1.14.1 Пространство имён

Intermech.Kernel.Search

5.1.14.2 Статические методы класса

FieldName

Статический метод, позволяющий получить название поля в таблицах СУБД для указанного обязательного атрибута.

Синтаксис:

```
string FieldName (ObligatoryObjectAttributes attr);
```

Параметры:

attr

Идентификатор обязательного атрибута.

Результат:

Название поля или **String.Empty**.

GetAttributeSourceType

Статический метод позволяет получить принадлежность обязательного атрибута к той или иной категории данных.

Синтаксис:

```
AttributeSourceTypes GetAttributeSourceType (ObligatoryObjectAttributes attribute);
```

Параметры:

attribute

Идентификатор обязательного атрибута.

Результат:

Какому источнику принадлежит указанный обязательный атрибут (объектам, связям, событиям и пр.).

GetCaption

Статический метод позволяет получить заголовок обязательного атрибута.

Синтаксис:

```
string GetCaption (ObligatoryObjectAttributes attr);
```

```
string GetCaption (string fieldName);
```

Параметры:

attr

Идентификатор обязательного атрибута.

fieldName

Название поля обязательного атрибута в таблицах СУБД.

Результат:

Заголовок обязательного атрибута.

GetDataType

Статический метод позволяет получить тип данных обязательного атрибута.

Синтаксис:

```
FieldTypes GetDataType (ObligatoryObjectAttributes attribute);
```

Параметры:

`attribute`

Идентификатор обязательного атрибута.

Результат:

Тип данных обязательного атрибута.

GetObligatoryObjectAttribute

Статический метод позволяет получить обязательный атрибут по его заголовку.

Синтаксис:

```
ObligatoryObjectAttributes GetObligatoryObjectAttribute (string caption);
```

Параметры:

`caption`

Заголовок обязательного атрибута.

Результат:

Идентификатор обязательного атрибута.

IsObligatoryObjectAttribute

Статический метод проверяет, является ли указанный заголовок заголовком обязательного атрибута.

Синтаксис:

```
bool IsObligatoryObjectAttribute (string caption);
```

Параметры:

`caption`

Заголовок атрибута.

Результат:

true, если указанный заголовок является заголовком обязательного атрибута.

5.1.15 Класс QueryConsts

Класс содержит константы, используемые в запросах к источникам данных.

5.1.15.1 Пространство имён

Intermech.Kernel.Search

5.1.15.2 Константы класса

const Int32 RecordCountOnly = 0

Только подсчитать количество записей, которое соответствует условиям запроса. В результате запрос вернет таблицу с одной строкой и колонкой, в которой будет содержаться количество записей.

const Int32 All = -1

Требуется прочитать все данные, соответствующие условию запроса.

const Int32 Default = -2

Требуется прочитать пакет данных с количеством записей, которое установлено в настройках системы (**IUserSession.MaxRows**).

const Int32 Hidden = -3

Зарезервировано для последующего использования.

5.1.16 Интерфейс IDeletable

Интерфейс реализуется объектами, которые допускают удаление.

5.1.16.1 Пространство имён

Intermech.Interfaces

5.1.16.2 Методы интерфейса

Delete

Метод удаляет реализующий объект.

Синтаксис:

```
Int32 Delete (Int64 deleteMode);
```

Параметры:

deleteMode

Используется для передачи различных флагов, управляющих режимами удаления. Набор флагов зависит от реализующего интерфейс объекта.

Результат:

Зависит от реализующего интерфейс объекта.

5.1.17 Интерфейс IDBGuid

Интерфейс, позволяющий получать глобальный уникальный идентификатор объекта, связи, атрибута, т.п.

5.1.17.1 Пространство имён

Intermech.Interfaces

5.1.17.2 Свойства интерфейса

Guid GUID { get; }

Глобальный уникальный идентификатор.

bool IsSystemGuid { get; }

Если равно **true**, то Guid является системным.

5.1.18 Интерфейс IDBSessionable

Интерфейс реализуется объектами, которые работают в рамках пользовательской или системной сессии.

5.1.18.1 Пространство имён

Intermech.Interfaces

5.1.18.2 Свойства интерфейса

UserSession Session { get; }

Интерфейс пользовательской или системной сессии.

5.1.19 Интерфейс IDBRecords

Интерфейс предназначен для получения из базы данных списка записей, отвечающих заданным условиям. Интерфейс унаследован от **IDBSessionable**.

5.1.19.1 Пространство имён

Intermech.Interfaces

5.1.19.2 Методы интерфейса

Delete

Метод удаляет элементы с идентификаторами idList и возвращает количество удаленных элементов. Если throwException равно **false**, то метод подавляет исключения и продолжает удалять оставшиеся элементы. Если throwException равно **true**, то удаление идет в одной транзакции. Параметр deleteMode передается в функцию Delete при удалении каждого элемента.

Синтаксис:

```
Int32 Delete (Int64[] idList, bool throwException, Int64 deleteMode);
```

Параметры:

idList

Список идентификаторов удаляемых данных (объектов, связей и пр.).

throwException

Если значение равно **true**, метод не будет подавлять исключения, но при возникновении исключения все выполненные операции будут отменены.

deleteMode

Режим удаления данных.

Результат:

Количество удалённых объектов.

Select

Возвращает таблицу данных в соответствии с параметрами запроса paramSet.

Синтаксис:

```
DataTable Select(DBRecordSetParams paramSet);
```

Параметры:

paramSet

Структура свойств, описывающих выборку записей из базы данных (см. п. [5.1.12](#)).

Результат:

Таблица с результатами запроса.

SetAttributesValues

Метод присваивает объектам или связям с идентификаторами idList значения атрибутов valuesList. Если throwException равно **false**, то метод подавляет исключения и продолжает присваивать значения атрибутам. Если throwException равно **true**, то присвоение идет в одной транзакции, но исключения не подавляются. Если addIfNotExists равно **true**, то метод добавляет несуществующие атрибуты, иначе генерирует исключение.

Внимание! Метод не поддерживает присвоение значений системным, двоичным и файловым атрибутам! Соответствующие объекты уже должны допускать изменение (быть взяты на изменение и пр.).

Синтаксис:

```
Int32 SetAttributesValues (Int64[] idList, AttributeValues[] valuesList, bool addIfNotExists, bool throwException);
```

Параметры:

idList

Список идентификаторов объектов или связей.

valuesList

Список значений атрибутов.

addIfNotExists

Если значение равно **true**, метод будет добавлять отсутствующие у объекта/связи атрибуты.

throwException

Если значение равно **true**, метод не будет подавлять исключения, но при возникновении исключения все выполненные операции будут отменены.

Результат:

Возвращается количество объектов или связей, которым успешно присвоились значения атрибутов.

5.1.20 Структура DocumentTypeSettings

Структура с информацией о дополнительных настройках типа документов: типа объектов, наследованного от типа объекта **Документы**. Для работы со структурой следует воспользоваться интерфейсом **IDocumentTypeSettingsService**.

5.1.20.1 Пространство имён

Intermech.Interfaces

5.1.20.2 Свойства структуры

DocumentFileExt

Расширение имени файла для типа документа (напр. “.txt”).

Синтаксис:

```
public string DocumentFileExt;
```

AdditionalDocumentFileExts

Список дополнительных расширений файлов для типа документа через разделитель «,» или «;» (напр. «.doc;.exe;.log»).

Синтаксис:

```
public string AdditionalDocumentFileExts;
```

OutputObjectTypes

Список уникальных идентификаторов (GUID) типов объектов через разделитель «,», выпускаемых по документам данного типа.

Синтаксис:

```
public string OutputObjectTypes;
```

DocumentTypeName

Наименование типа документов, возможно наследованное (при чтении структуры).

Синтаксис:

```
public string DocumentTypeName;
```

DocumentTypeCode

Наименование кода типа документов, возможно наследованное (при чтении структуры).

Синтаксис:

```
public string DocumentTypeCode;
```

DocumentNameInStamp

Отрисовывать наименование типа в штампе чертежа.

Синтаксис:

```
public bool DocumentNameInStamp;
```

DocumentTypeCodeInDesignation

Код типа документа входит в обозначение.

Синтаксис:

```
public bool DocumentTypeCodeInDesignation;
```

5.1.21 Интерфейс IDocumentTypeSettingsService

Интерфейс для управления дополнительными настройками типа документов: типа объектов, наследованного от типа объекта **Документы**.

5.1.21.1 Пространство имён

Intermech.Interfaces

5.1.21.2 Способы получения интерфейса

На серверной стороне интерфейс можно получить у статического контейнера сервисов **Intermech.Interfaces.Server.ServerServices**:

```
ServerServices.GetService(typeof(IDocumentTypeSettingsService)) as IDocumentTypeSettingsService;
```

На клиенте интерфейс может быть получен у сессии с помощью метода **IUserSession.GetCustomService**.

5.1.21.3 Методы интерфейса

GetDocumentTypesByFileExt

Метод позволяет получить список типов документов, использующих заданное расширение файла.

Синтаксис:

```
int[] GetDocumentTypesByFileExt (Guid sessionGuid, string fileExt);
```

Параметры:

sessionGuid

Уникальный идентификатор сессии.

fileExt

Расширение файла, напр «.txt».

Результат:

Список типов документов, использующих заданное расширение файла.

GetDocumentTypesByOutputObjectTypes

Метод позволяет получить список типов документов, у которых в качестве выпускаемого указан хотя бы один из перечисленных типов объектов.

Синтаксис:

```
int[] GetDocumentTypesByOutputObjectTypes (Guid sessionGuid, int[] outputObjectTypes, int rootDocumentObjectType);
```

Параметры:

sessionGuid

Уникальный идентификатор сессии.

outputObjectTypes

Список выпускаемых типов объектов, по которым следует выполнять поиск.

`rootDocumentObjectType`

Корневой тип документа, относительно которого следует выполнять поиск по дереву типов документов. Корневой тип документа в иерархии типов объектов не может находиться выше типа объектов **Документы**.

Результат:

Список типов документов, у которых в качестве выпускаемого указан хотя бы один из перечисленных в **outputObjectTypes** типов объектов.

GetSettings

Метод позволяет получить дополнительные настройки типа документа.

Синтаксис:

```
DocumentTypeSettings GetSettings (Guid sessionGuid, int documentType);
```

Параметры:

`sessionGuid`

Уникальный идентификатор сессии.

`documentType`

Идентификатор типа документа.

Результат:

Структура `DocumentTypeSettings` с дополнительными настройками типа документа.

InheritedFromConstructorDocuments

Метод позволяет проверить наследование типа объектов от типа документов **Конструкторские документы**.

Синтаксис:

```
bool InheritedFromConstructorDocuments (Guid sessionGuid, int documentType);
```

Параметры:

`sessionGuid`

Уникальный идентификатор сессии.

`documentType`

Идентификатор типа объекта.

Результат:

true, если тип объекта `documentType` наследован от типа “Конструкторские документы”.

InheritedFromDocuments

Метод позволяет проверить наследование типа объектов от типа документов “Документы”.

Синтаксис:

```
bool InheritedFromDocuments (Guid sessionGuid, int documentType);
```

Параметры:

`sessionGuid`

Уникальный идентификатор сессии.

`documentType`

Идентификатор типа объекта.

Результат:

true, если тип объекта `documentType` наследован от типа **Документы**.

SetSettings

Метод позволяет установить дополнительные настройки типа документа.

Синтаксис:

```
void SetSettings (Guid sessionGuid, int documentType, DocumentTypeSettings documentTypeSettingsData);
```

Параметры:

`sessionGuid`

Уникальный идентификатор сессии.

`documentType`

Идентификатор типа документа.

`documentTypeSettingsData`

Структура **DocumentTypeSettings** с инициализированными дополнительными настройками типа документа.

5.2 Права доступа к объектам

Для каждого информационного объекта и метаданного в системе можно настроить права доступа. Для управления правами доступа в ядре IPS Development Kit доступно несколько интерфейсов, классов и перечислителей.

5.2.1 Перечислитель ActionCategory

Категория действия (чтение, модификация, администрирование) – предназначена для группирования действий в диалогах назначения прав доступа.

5.2.1.1 Пространство имён

Intermech.Interfaces

5.2.1.2 Значения перечислителя

NotDefined = 0

Категория не определена.

Read = 1

Чтение и просмотр данных.

Write = 2

Модификация данных.

Admin = 3

Администрирование.

По умолчанию для объектов права доступа из категорий **Read** и **Write** разрешены всем пользователям, а **Admin** – только роли **Администратор**. Для метаданных группе **ВСЕ_ПОЛЬЗОВАТЕЛИ** разрешены только права из категории **Read**. Существуют программные методы переопределения умолчаний для определенных типов объектов и прав доступа.

5.2.2 Перечислитель AccessType

Перечислитель описывает права доступа к чему-либо.

5.2.2.1 Пространство имён

Intermech.Interfaces

5.2.2.2 Значения перечислителя

Default = 0

Права доступа по умолчанию.

NoGrant = 1

Права доступа не разрешены.

Grant = 2

Права доступа разрешены.

Deny = 3

Приоритетный запрет прав доступа.

GrantAlways = 4

Приоритетное разрешение прав доступа.

При сложении различных типов прав доступа (например, если права назначены нескольким группам, в которые входит данный пользователь) результат проверки прав будет следующий:

NoGrant + Grant = права разрешены;

Grant + Deny = права запрещены;

Deny + GrantAlways = права разрешены;

NoGrant + GrantAlways = права разрешены.

5.2.3 Перечислитель ActionType

Перечислитель описывает тип действия, для которого проверяются права доступа.

5.2.3.1 Пространство имён

Intermech.Interfaces

5.2.3.2 Значения перечислителя

Применение значений перечислителя зависит от типа объектов и метаданных. Например, значения **GetAccess** и **SetAccess** применяются для проверки прав на чтение и изменение прав доступа. Список возможных прав доступа содержится в свойстве **AccessActions** класса **DBSessionable**, который является базовым для всех классов, способных осуществлять проверку прав доступа от имени сессии (см. п. 9.3.3). Список допустимых прав доступа можно пополнять в методе **DBSessionable.InitSecurityOptions**.

5.2.4 Перечислитель CheckAccessFlags

Перечислитель (набор битовых флажков, [Flags()]) описывает набор флажков, управляющих проверкой прав доступа.

5.2.4.1 Пространство имён

Intermech.Interfaces

5.2.4.2 Значения перечислителя

None = 0

Нет проверки прав доступа.

ThrowACEException = 1

Генерировать исключение **AccessDeniedAxception** в случае отсутствия прав доступа.

BatchCheck = 2

Режим пакетной проверки. Используется в случае, если несколько проверок прав доступа должны производиться совместно.

5.2.5 Структура ActionProperties

Структура описывает действие, которое можно применить к объекту или категории данных.

5.2.5.1 Пространство имён

Intermech.Interfaces

5.2.5.2 Свойства структуры

ActionID

Идентификатор действия.

Синтаксис:

```
public ActionType ActionID;
```

Category

Категория действия (чтение, изменение, администрирование).

Синтаксис:

```
public ActionCategory Category;
```

ConnectedActions

Задаёт список связанных действий, права доступа для которых нужно снимать при снятии прав для опции ActionID. Если таковых нет, **ConnectedActions = null**.

Синтаксис:

```
public ActionType[] ConnectedActions;
```

DefaultAccess

Разрешено ли право доступа по умолчанию.

Синтаксис:

```
public bool DefaultAccess;
```

Name

Описание действия.

Синтаксис:

```
public string Name;
```

5.2.6 Интерфейс IDBSecurity

Интерфейс предназначен для проверки и назначения прав доступа объектам и метаданным. Поддерживается базовым обработчиком объектов **DObject**, а также большинством объектов, реализующих работу с метаданными, описанными в главе 2 настоящего руководства.

5.2.6.1 Пространство имён

Intermech.Interfaces

5.2.6.2 Свойства интерфейса

CategoryDescriptor Descriptor { get; }

Возвращает идентификационную структуру данного объекта, используемую для хранения информации о правах доступа для него.

bool IsAccessTypeDeny { get; }

Возвращает **true**, если последняя проверка прав доступа завершилась приоритетным запретом.

bool IsLastDefault { get; }

Возвращает **true**, если последняя проверка прав доступа вернула права доступа по умолчанию.

string ObjectName { get; }

Возвращает имя данного объекта для записи его в различные логи и сообщения.

5.2.6.3 Методы интерфейса

CheckAccess

Проверяет права доступа.

Синтаксис:

```
bool CheckAccess(ActionType rightID);
```

```
bool CheckAccess(ActionType rightID, bool defaultAccess);
```

```
bool CheckAccess(ActionType rightID, bool defaultAccess, bool aThrowACEException);
```

```
bool CheckAccess(ActionType rightID, bool defaultAccess, CheckAccessFlags flags);
```

Параметры:

rightID

Проверяемые права доступа.

defaultAccess

Права доступа по умолчанию. Это значение будет использовано в качестве результата функции, если при вычислении не будет обнаружено явных назначений прав доступа данному пользователю (а также его группам и ролям) на данный объект. Если свойство не задано, то значение по умолчанию зависит от категории прав доступа, а также от реализующего проверку прав объекта.

aThrowACEException

Генерировать ли исключение **AccessDeniedException**, если нет прав доступа.

flags

Флаги, управляющие проверкой прав доступа.

Результат:

true, если у текущего пользователя есть указанные права доступа.

GetAccessList

Возвращает таблицу прав доступа, назначенных указанному объекту. Для выполнения данного метода требуются права доступа **ActionType.GetAccess**.

Синтаксис:

```
DataTable GetAccessList(out ActionProperties[] actions, out QuickObjectInfo[] users);
```

Параметры:

actions

Возвращает массив описателей допустимых действий, на которые можно назначать права доступа для данного объекта.

users

Возвращает массив с описателями пользователей, групп и ролей, которым назначены права доступа на данный объект. Используется для быстрого отображения назначений в окне просмотра и назначения прав доступа.

Результат:

Возвращает **DataTable** с правами доступа. В расширенных свойствах таблицы атрибут **ReadOnly = 1** говорит о том, что данный пользователь права доступа менять не имеет права.

Обязательные атрибуты	Тип поля в СУБД	Примечания	Свойства в IDBSecurity

F_KEY	INTEGER	Уникальный идентификатор записи в таблице прав доступа.	
F_CATEGORY_TYPE	INTEGER	Идентификатор категории. Системные идентификаторы объявлены в классе Intermech.Interfaces.Consts . Например, для версий информационных объектов это CategoryObjectVersion .	Descriptor.CategoryType
F_CATEGORY_ID	INTEGER	Идентификатор объекта данной категории, которому назначены права доступа.	Descriptor.CategoryID
F_RIGHT_ID	INTEGER	Идентификатор действия, на которое назначено право доступа (значение из перечислителя ActionType).	
F_USER_ID	INTEGER	Идентификатор пользователя, группы или роли, которой назначено право доступа (в качестве идентификатора используется ObjectID соответствующего объекта).	
F_PARENT_KEY	INTEGER	Содержит константу Consts.DefaultAccessKey для записей, содержащих права доступа по умолчанию. Для остальных записей содержит 0.	
F_RIGHT_TYPE	INTEGER	Право доступа (содержит значение из перечислителя AccessType)	
F_OWNER_ID	INTEGER	Идентификатор пользователя, назначившего право доступа.	
F_BEGIN_DATE	DATETIME	Дата начала действия права доступа. Содержит DBNull.Value если право действует постоянно.	
F_END_DATE	DATETIME	Дата завершения действия права доступа. Содержит DBNull.Value если право действует постоянно.	

GetRelatedSecurity

Получить список связанных интерфейсов безопасности, права которых также нужно показывать и администрировать вместе с правами данного объекта.

Синтаксис:

```
IDBSecurity[] GetRelatedSecurity();
```

Результат:

Список связанных интерфейсов безопасности.

RestoreAdminAccess

Метод восстанавливает права роли **Администратор** на данный объект. Функционирует только если текущая сессия работает под ролью **Администратор**.

Синтаксис:

```
void RestoreAdminAccess();
```

SetAccess

Метод позволяет назначить права доступа. Для выполнения данного метода требуются права доступа **ActionType.SetAccess**.

Синтаксис:

```
void SetAccess(DataTable accessList, params object[] AddInfo);
```

Параметры:

accessList

Таблица с правами доступа, аналогичная по формату таблице, возвращаемой методом **GetAccess**. Права по умолчанию в данную таблицу включать не следует. Для новых записей о правах доступа поле **F_KEY** должно содержать значение 0. Для удаляемых записей поле **F_RIGHT_TYPE** должно содержать константу **Consts.DeleteRecord**.

AddInfo

Параметр зарезервирован для будущего использования.

5.2.7 Класс DBSessionable

Базовый абстрактный класс, в котором содержится ссылка на текущую сессию и механизм управления правами доступа этой сессии к реализуемому объекту. Все классы серверного ядра IPS Development Kit, реализующие интерфейс **IDBSecurity**, унаследованы от данного класса или от его потомков.

5.2.7.1 Пространство имён

Intermech.Kernel

Дополнительно к свойствам и методам, описанным в интерфейсе **IDBSecurity**, класс содержит следующие свойства и методы:

5.2.7.2 Свойства класса

ListDictionary AccessActions { get; }

Содержит список действий, на которые можно назначить права доступа, в виде **Идентификатор действия** = **права по умолчанию для группы ВСЕ_ПОЛЬЗОВАТЕЛИ**. В реализации по умолчанию список содержит действия **ActionType.GetAccess=false** и **ActionType.SetAccess=false**. Пополнять список можно в методе **InitSecurityOptions**, который вызывается из базового конструктора класса.

virtual Int64 AccessOwnerID { get; }

Возвращает идентификатор владельца данного объекта. Позволяет корректно вычислять права доступа, предоставленные группе **ВЛАДЕЛЕЦ_ОБЪЕКТА**.

protected virtual bool Deleted { get; set; }

Возвращает **true**, если объект, который представлен данным экземпляром класса, уже удален из базы данных. Позволяет определить, что данным экземпляром пользоваться уже нельзя.

virtual string ObjectName { get; }

Возвращает имя данного объекта для записи его в различные логи и сообщения.

UserSession UserSession { get; }

Содержит ссылку на пользовательскую сессию, в рамках которой созданы экземпляр данного класса.

5.2.7.3 Методы класса

InitSecurityOptions

Метод должен быть обязательно вызван из конструктора унаследованного объекта для инициализации полей, необходимых для работы с правами доступа.

protected virtual void InitSecurityOptions (Int32 aCategoryType, Int64 aCategoryID);

Параметры:

aCategoryType

Идентификатор категории, к которой принадлежит данный объект (например, атрибуты, типы связей и объектов, версии объектов и т.п.). Идентификаторы системных категорий описаны в классе **Intermech.Interfaces.Consts**. Они начинаются с префикса **Category**. Например, типы связей имеют идентификатор категории **Consts.CategoryRelationType**. Допускается вводить собственные категории данных, но их идентификаторы не должны быть меньше 10000. Также следует добавить название своей категории в словарь **Intermech.Interfaces.CategoryNames** (например, **Consts.CategoryNames[MyCategoryID] = "Моя категория"**).

aCategoryID

Идентификатор объекта данной категории. Например, для типов связей должен содержать идентификатор типа связей.

При перекрытии данного метода сперва следует вызвать его базовую реализацию.

5.3 Стандартные информационные объекты

Информационный объект – базовая единица информации, описывающая какой-либо материальный или виртуальный объект, действие или явление. Каждый информационный объект характеризуется атрибутами, связями с другими объектами и своим жизненным циклом. Допустимый набор атрибутов, связей и шагов ЖЦ определяется типом объекта. Объекты бывают версионными и неверсионным, что также определяется типом объектов. Неверсионные объекты могут иметь только одну версию, которая собственно и является самим информационным объектом. Версионный объект представляет собой совокупность всех версий объекта, объединённых общим идентификатором объекта.

5.3.1 Интерфейс IDBObject

Интерфейс общего обработчика информационного объекта. Интерфейс унаследован от интерфейсов **IDBAttributable**, **IDBSessionable**.

5.3.1.1 Пространство имён

Intermech.Interfaces

5.3.1.2 Способы получения интерфейса

`IUserSession.GetObject(Int64 objectID);`

`IUserSession.GetObject(Int64 objectID, bool throwNotFoundException);`

`IUserSession.GetObject(Guid objectGUID);`

`IUserSession.GetObject(Guid objectGUID, bool throwNotFoundException);`

Возвращает обработчик версии объекта по его локальному или глобальному идентификатору.

Параметры:

`objectID`

Локальный идентификатор версии объекта.

`objectGUID`

Глобальный идентификатор версии объекта.

`throwNotFoundException`

Если параметр равен **true** (а также при отсутствии у функции данного параметра), то система генерирует исключение при отсутствии в базе данных запрошенной версии объекта. При значении **false** функция возвращает **null**, если указанная версия объекта в базе не найдена.

`IUserSession.GetObjectActualCopy(Int64 objectID, bool throwNotFoundException);`

Возвращает обработчик рабочей копии объекта для данного пользователя (если эта версия взята данным пользователем на изменение). Если рабочей копии у объекта нет (или объект взят на изменение другим пользователем), то метод возвращает обработчик архивной копии объекта.

Параметры:

`objectID`

Локальный идентификатор рабочей или архивной копии объекта.

`IUserSession.GetObjectBaseVersionByID(Int64 id, bool throwNotFoundException);`

Возвращает обработчик базовой версии объекта.

Параметры:

`id`

Идентификатор объекта.

`IUserSession.GetObjectByVersionRule(Int64 id, string FiltrationSettings, bool throwNotFoundException);`

`IUserSession.GetObjectByVersionRule(Guid guid, string FiltrationSettings, bool throwNotFoundException);`

Возвращает обработчик версии объекта, соответствующей указанному правилу подбора версий.

Параметры:

`id`

Локальный идентификатор объекта.

`guid`

Глобальный идентификатор объекта.

`FiltrationSettings`

Идентификатор настроек фильтрации состава. Если требуется применить текущие настройки пользователя, то в качестве значения следует передать строку `SystemGUIDs.filtrationUserDefault`.

5.3.1.3 Свойства интерфейса

Caption

Возвращает заголовок версии объекта –строковое представление объекта для его отображения в навигаторе объектов.

Синтаксис:

`string Caption { get; set; }`

CheckoutBy

Идентификатор пользователя, взявшего версию объекта на изменение. Содержит 0 для версий, не взятых на изменение.

Синтаксис:

```
Int64 CheckoutBy { get; }
```

CreateDate

Дата и время создания объекта.

Синтаксис:

```
DateTime CreateDate { get; }
```

FiltrationState

Статус последней фильтрации текущего объекта по правилу подбора версий.

Синтаксис:

```
ObjectFiltrationState FiltrationState { get; set; }
```

GUID

Глобальный идентификатор объекта (один для всех версий объекта).

Синтаксис:

```
Guid GUID { get; set; }
```

ID

Уникальный идентификатор объекта (только для чтения). Содержит одинаковое значение у всех версий данного объекта.

Синтаксис:

```
Int64 ID { get; }
```

IsCreationMode

Если значение свойства равно **true**, то это заготовка версии объекта.

Синтаксис:

```
bool IsCreationMode { get; }
```

LCStep

Идентификатор этапа жизненного цикла версии объекта.

Синтаксис:

```
int LCStep { get; set; }
```

ModificationID

Номер группы изменений (только для чтения). Инициализируется при включении версии объекта в контекст редактирования. Иначе содержит 0.

Синтаксис:

```
Int64 ModificationID { get; set; }
```

ModifyDate

Дата последней модификации версии объекта (только для чтения). Дата автоматически обновляется при изменении любого атрибута у данной версии объекта.

Синтаксис:

```
DateTime ModifyDate { get; }
```

NameInMessages

Возвращает строку для отображения имени объекта в информационных сообщениях.

Синтаксис:

```
string NameInMessages { get; }
```

ObjectGUID

Глобальный идентификатор данной версии объекта. Идентификатор можно изменять только если включен режим разработчика **IUserSession.DeveloperMode** (см. п. [2.4.2](#)).

Синтаксис:

```
Guid ObjectGUID { get; set; }
```

ObjectID

Локальный идентификатор данной версии объекта (только для чтения).

Синтаксис:

```
Int64 ObjectID { get; }
```

ObjectModifyMode

Возвращает способ модификации объекта на текущем шаге ЖЦ.

Синтаксис:

```
ObjectModifyModes ObjectModifyMode { get; }
```

ObjectType

Идентификатор типа объекта.

Синтаксис:

```
int ObjectType { get; set; }
```

ObjectVerType

Статус версии (-1 – заготовка, 0 – версия объекта, 1 – версия находится в процессе импорта из портфеля или web-портала). Значения соответствуют перечислителю **Intermech.ObjectRecordKind**.

Синтаксис:

```
int ObjectVerType { get; set; }
```

OwnerID

Идентификатор владельца объекта. По умолчанию владельцем версии объекта становится пользователь, который ее создал. Таким образом в этом поле содержится **ObjectID** пользователя, который создал данную версию объекта. Для изменения владельца версии объекта у текущего пользователя должны быть права **ActionType.TakeOwnership**.

Синтаксис:

```
Int64 OwnerID { get; set; }
```

ParentVersionID

Идентификатор версии объекта, на основе которой была создана данная версия объекта. Если это самая первая версия (или родительская версия была удалена), то свойство возвращает -1.

Синтаксис:

```
Int64 ParentVersionID { get; }
```

ProjectID

Идентификатор проекта, к которому принадлежит объект. Если равно **0**, то объект создан вне контекста проекта.

Синтаксис:

```
Int64 ProjectID { get; set; }
```

ReadOnly

Возвращает **true**, если версия объекта не может быть модифицирована в данное время.

Синтаксис:

```
bool ReadOnly { get; }
```

SiteID

Символьные идентификаторы узлов информационной сети. Первый символ – идентификатор узла, в котором была создана данная версия объекта. Второй символ – идентификатор узла, владеющего данной версией объекта. Если свойство содержит пустую строку, то значит эта версия объекта была создана на данном узле (базе данных) и никогда не публиковалась на портале для передачи на другие узлы сети.

Синтаксис:

```
string SiteID { get; }
```

SubjectAreas

Строка символов, определяющих предметные области, к которым относится объект данного типа. Если строка пустая, то объект относится ко всем областям.

Синтаксис:

```
string SubjectAreas { get; }
```

VersionID

Порядковый номер версии объекта (только для чтения). Генерируется системой автоматически начиная с 0.

Синтаксис:

```
int VersionID { get; }
```

5.3.1.4 Методы интерфейса

CancelChanges

Возвращает версию объекта в базу с отменой изменений в рабочей копии.

Синтаксис:

```
int CancelChanges();  
int CancelChanges(bool isAdminMode);
```

Параметры:

isAdminMode

Если значение равно **true** и пользователь зашел под ролью Администратор, то система позволяет отменять изменения в объектах, взятых на изменение другими пользователями.

Результат:

Зарезервировано.

CheckEdit

Проверяет можно ли редактировать свойства данного объекта (если нельзя, то генерируется исключение).

Синтаксис:

```
void CheckEdit();
```

CheckIn

Возвращает версию объекта в базу с сохранением изменений.

Синтаксис:

```
int CheckIn();
```

Результат:

Зарезервировано.

CheckOut

Берет версию объекта из базы на изменение.

Синтаксис:

```
IDBObject CheckOut();
```

```
IDBObject CheckOut(bool throwModifyModeException);
```

Параметры:

throwModifyModeException

Если значение равно **true**, то при попытке взять на изменение объект, чей шаг не допускает взятие на изменение, метод генерирует исключение.

Результат:

Возвращает обработчик версии объекта, которая доступна для изменения. Это может быть рабочая копия данной версии объекта, новая версия объекта или ссылка на этот же объект (все зависит от поведения данного типа объекта на данном шаге жизненного цикла).

CommitCreation

Метод завершает создание версии объекта, переводя заготовку в объект.

Синтаксис:

```
void CommitCreation(bool deleteOnException);
```

```
void CommitCreation(bool deleteOnException, bool autoCheckOut);
```

Параметры:

deleteOnException

Если значение равно **true**, то при возникновении исключения заготовка объекта будет удалена данной функцией автоматически. Иначе заготовку нужно будет либо перевести в объект повторным вызовом данной функции, либо удалить методом **Delete**.

autoCheckOut

Нужно ли автоматически брать на изменение объект в случае, если на первом шаге ЖЦ изменение объекта может производиться только через рабочую копию. Если параметр равен **true**, то функция вернет рабочую копию созданной версии объекта.

Delete

Удаляет версию объекта (если текущий шаг схемы ЖЦ позволяет перевод объекта на шаг **Удалено**). Если в настройках данного типа объектов указано ненулевое время жизни удаленных объектов, то версия переводится на шаг жизненного цикла **Удалено**.

Синтаксис:

```
int Delete(Int64 DeleteMode);
```

Параметры:

DeleteMode

Параметр зарезервирован для будущего использования.

Результат:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

GetHashFile

Возвращает хэш данной версии объекта, который используется для создания и проверки электронных подписей объекта.

Синтаксис:

```
string GetHashFile();  
string GetHashFile(int versionID);
```

Параметры:

versionID

Номер версии алгоритма получения данного файла.

Результат:

Строка с хэш-кодом данной версии объекта.

GetHashVersion

Возвращает номер текущей версии алгоритма генерации хэш-файла функцией GetHashFile().

Синтаксис:

```
int GetHashVersion();
```

Результат:

Номер текущей версии алгоритма генерации хэш-файла.

GetLCHistory

Возвращает таблицу с историей изменения шага жизненного цикла данной версии объекта.

Синтаксис:

```
DataTable GetLCHistory();  
DataTable GetLCHistory(bool allVersions);
```

Параметры:

allVersions

Если значение равно **true**, то возвращает историю всех версий данного объекта.

Результат:

Таблица с историей изменения шага жизненного цикла данной версии объекта.

Имя поля	Тип поля	Комментарий
F_KEY	BIGINT	Уникальный номер записи в данной таблице.
F_OBJECT_ID	BIGINT	Идентификатор версии объекта.
F_LC_STEP	INTEGER	Идентификатор шага ЖЦ, на который перевели версию объекта.
F_START_DATE	DATETIME	Дата и время перевода версии объекта.

isParentType

Проверяет тип данной версии объекта на принадлежность указанной ветке дерева типов объектов.

Синтаксис:

```
bool isParentType(Guid guid);
```

Параметры:

guid

Глобальный идентификатор типа объекта.

Результат:

Возвращает значение **true**, если тип данной версии объекта унаследован от типа с глобальным идентификатором **guid** или сам тип объекта имеет идентификатор **guid**.

SaveChanges

Сохраняет изменения файловой копии объекта в рабочую копию (которая находится в базе данных).

Синтаксис:

```
void SaveChanges().  
void SaveChanges(bool saveToArcCopy).  
saveToArcCopy
```

Если **true**, то метод также сохраняет изменения рабочей копии в архивную копию объекта.

5.3.2 Класс DBObject

Базовый класс для всех информационных объектов. Реализует интерфейсы **IDBObject**, **IDBGuid**, **IDeletable**, **IDBLifecycleLevel**, **IDBSecurityCollection**. От данного класса наследуются обработчики для определённых типов объектов. Сам класс унаследован от объекта **DBAttributable**, который реализует набор функций по работе с атрибутами.

5.3.2.1 Пространство имён

Intermech.Kernel

5.3.2.2 Свойства класса

AccessOwnerID

Идентификатор объекта-владельца.

Синтаксис:

```
public override Int64 AccessOwnerID { get; }
```

Attributes

Коллекция атрибутов объекта. Зачитывается из БД при первом обращении. Если нужно получить или модифицировать только один атрибут, следует обращаться к нему не через данную коллекцию, а через функции `GetAttributeByID`, `GetAttributeByName`, `GetAttributeByGuid`.

Синтаксис:

```
public override IDBAttributeCollection Attributes { get; }
```

IsSystemGUID

Если значение свойства равно **true**, то данный глобальный идентификатор является системным.

Синтаксис:

```
public bool IsSystemGUID { get; }
```

ObjectTypeClass

Возвращает интерфейс обработчика объектов данного типа.

Синтаксис:

```
public IDBObjectType ObjectTypeClass { get; }
```

TypeID

Идентификатор типа объектов, к которым относится данная версия объектов.

Синтаксис:

```
public int TypeID { get; }
```

5.3.2.3 Методы класса

AfterSetLCStep

Метод вызывается после перевода объекта на новый шаг ЖЦ.

Синтаксис:

```
protected virtual void AfterLCStep();
```

DoAfterCreate

Метод вызывается после создания заготовки объекта. Если метод генерирует исключение, то создание объекта прекращается, а его заготовка удаляется откатом транзакции.

Синтаксис:

```
public virtual void DoAfterCreate();
```

DoAfterCreateRelation

Метод вызывается в транзакции сразу после добавления в состав данной версии объекта связи `newRelation`.

Синтаксис:

```
protected virtual void DoAfterCreateRelation(IDBRelation newRelation);
```

DoBeforeCommitCreation

Метод вызывается из метода `CommitCreation` и позволяет унаследованным объектам выполнить действия перед завершением создания версии объекта (например, проверить заполнение атрибутов и отменить операцию `CommitCreation`).

DoBeforeMakeBaseVersion

Метод вызывается перед тем, как данную версию объекта сделают базовой. Параметр `flags` зарезервирован для будущего использования.

Синтаксис:

```
protected virtual void DoBeforeMakeBaseVersion(int flags);
```

Синтаксис:

```
protected virtual void DoBeforeCommitCreation();
```

DoCommitCreation

Метод вызывается из метода CommitCreation и позволяет унаследованным объектам выполнить действия по завершению создания версии объекта.

Синтаксис:

```
protected virtual void DoCommitCreation();
```

DoDelete

Метод вызывается в транзакции в момент удаления версии объекта. Позволяет, например, проверить условия удаления и отменить данную операцию.

Синтаксис:

```
protected virtual void DoDelete();
```

DoPurge

Метод вызывается в транзакции перед уничтожением в БД данной версии объекта (даже если это заготовка или рабочая копия объекта).

Синтаксис:

```
protected virtual void DoPurge(Int64 DeleteMode);
```

5.4 Работа с проектами

Для работы с объектами типа **Проекты** в ядре системы есть стандартные интерфейсы, классы и перечислители.

5.4.1 Перечислитель ProjectFiltrationModes

Перечислитель описывает способ фильтрации списков объектов в зависимости от их принадлежности к проектам.

5.4.1.1 *Пространство имён*

Intermech.Interfaces

5.4.1.2 *Значения перечислителя*

None = 0

Не фильтровать объекты.

CurrentProject = 1

Показывать объекты только текущего проекта.

UserProjects = 2

Показывать объекты проектов, в которых участвует текущий пользователь.

5.4.2 Интерфейс IDBProjectObject

Интерфейс обработчика информационного объекта типа «Проекты».

5.4.2.1 *Пространство имён*

Intermech.Interfaces.Projects

5.4.2.2 *Свойства интерфейса*

LinkedObjects

Список объектов, принадлежащих проекту.

Синтаксис:

```
Int64[] LinkedObjects { get; }
```

LinkedObjectsCount

Количество объектов, принадлежащих проекту.

Синтаксис:

```
Int64 LinkedObjectsCount { get; }
```

5.4.2.3 Методы интерфейса

ExcludeParticipants

Исключает указанных пользователей из состава участников проекта. Для выполнения данной операции текущий пользователь должен быть менеджером проекта.

Синтаксис:

```
void ExcludeParticipants (Int64[] users);
```

Параметры:

users

Список идентификаторов пользователей, которых надо исключить из состава участников проекта.

GetParticipants

Метод возвращает массив с информацией по участникам проекта.

Синтаксис:

```
ProjectParticipantInfo[] GetParticipants ();
```

Результат:

Массив с информацией по участникам проекта.

GetParticipantsInfo

Метод возвращает массив расширенной информацией по участникам проекта.

Синтаксис:

```
ProjectParticipantInfoEx[] GetParticipantsInfo ();
```

Результат:

Массив с расширенной информацией по участникам проекта.

IncludeParticipants

Добавляет указанных пользователей в состав участников проекта.

Синтаксис:

```
void IncludeParticipants (ProjectParticipantInfo[] participants);
```

Параметры:

participants

Список добавляемых участников проекта. Для выполнения данной операции текущий пользователь должен быть менеджером проекта.

IsProjectManager

Метод проверяет, является ли указанный пользователь (или пользователь текущей сессии) менеджером проекта.

Синтаксис:

```
bool IsProjectManager ();
```

```
bool IsProjectManager (Int64 userID);
```

Параметры:

userID

Идентификатор пользователя.

Результат:

true, если пользователь является менеджером проекта.

IsProjectParticipant

Метод проверяет, является ли указанный пользователь (или пользователь текущей сессии) участником проекта.

Синтаксис:

```
bool IsProjectParticipant ();
```

```
bool IsProjectParticipant (Int64 userID);
```

Параметры:

userID

Идентификатор пользователя.

Результат:

true, если пользователь является участником проекта.

5.4.3 Класс ProjectParticipantInfo

Класс предназначен для хранения информации об участнике проекта.

5.4.3.1 Пространство имён

Intermech.Interfaces.Projects

5.4.3.2 Поля класса

ParticipantID

Идентификатор участника проекта.

Синтаксис:

```
public Int64 ParticipantID;
```

ProjectManager

Если true, то указанный участник является менеджером проекта.

Синтаксис:

```
public bool ProjectManager;
```

5.4.4 Класс ProjectParticipantInfoEx

Класс предназначен для хранения информации об участнике проекта, включая имя пользователя. Класс унаследован от класса **ProjectParticipantInfo**.

5.4.4.1 Пространство имён

Intermech.Interfaces.Projects

5.4.4.2 Поля класса

Caption

Имя участника проекта.

Синтаксис:

```
public string Caption;
```

5.5 Создание и поиск объектов

5.5.1 Интерфейс IDBObjectCollection

Интерфейс позволяет создавать новые объекты, получать список уже существующих объектов, проверять существование объектов с указанными идентификаторами и пр. Реализация осуществляется объектом **IDBObjectCollection**. Интерфейс унаследован от **IDBRecords**.

5.5.1.1 Пространство имён

Intermech.Interfaces

5.5.1.2 Способы получения интерфейса

Для получения коллекции объектов типа objectType и всех дочерних типов используются методы интерфейса IUserSession. Под коллекцией объектов в данном случае понимается объект, управляющий списком объектов.

IUserSession.GetObjectCollection(int objectType):

Параметры:

objectType

Идентификатор типа объектов. Если значение параметра objectType равно -1, то метод возвращает коллекцию объектов всех типов.

```
IUserSession.GetObjectCollection(Guid objectTypeGuid);
```

Параметры:

objectTypeGuid

Глобальный идентификатор типа объектов.

5.5.1.3 Свойства интерфейса

ObjectTypeID

Тип объектов, обрабатываемых коллекцией.

Синтаксис:

```
int ObjectTypeID { get; set; }
```

5.5.1.4 Методы интерфейса

ActualDateObjectVersion

Метод отыскивает версию объекта, которая существовала на указанную дату на указанном уровне продвижения или на указанном шаге жизненного цикла (значения даты и уровня/шага ЖЦ находятся в правиле подбора версий). Если версия не найдена, вернётся значение **-1**.

Синтаксис:

```
Int64 ActualDateObjectVersion (Int64 objectID, VersionsRule rule, out ObjectFiltrationState state);
```

Параметры:

objectID

Идентификатор версии объекта.

rule

Правило подбора версий, в которое включены:

- дата актуального состава,
- уровень продвижения или шаг жизненного цикла.

state

В данной переменной будет возвращён статус найденной версии.

Результат:

Найденный идентификатор версии объекта или **- 1**.

AddAttribute

Добавляет атрибут с идентификатором attributeID версиям объектов objectIDs и присваивает им значения values (если значение values не пустое). Если значение ignoreExceptions равно **true**, а данный атрибут нельзя добавить к данному объекту, то пытается добавить атрибут к следующему объекту в списке. Если атрибут у объекта существует, то ему просто присваивается значение из параметра values.

Синтаксис:

```
CommandResult AddAttribute(Int64[] objectIDs, object attributeID, object[] values, bool ignoreExceptions);
```

Параметры:

objectIDs

Массив идентификаторов версий объектов.

attributeID

Идентификатор атрибута.

values

Значения, присваиваемые объектам.

ignoreExceptions

Определяет, нужно ли генерировать исключение при попытке добавления запрещенного атрибута.

Результат:

Структура с результатами выполнения метода.

Create

Создает заготовку объекта такого же типа, как и тип, указанный при получении IDBObjectCollection.

Синтаксис:

```
IDBObject Create();
```

Результат:

Обработчик созданной заготовки объекта.

Create

Создает заготовку нового объекта на основе прототипа prototype. Если коллекция типизированная, то создается объект того же типа, что и коллекция, иначе тип берется у прототипа.

Синтаксис:

```
IDBObject Create(IDBObject prototype);
```

Параметры:

prototype

Объект-прототип.

Результат:

Обработчик созданной заготовки объекта.

Create

Создает заготовку нового объекта типа objectType.

Синтаксис:

```
IDBObject Create(Int32 objectType);
```

Параметры:

objectType

Идентификатор типа объекта.

Результат:

Обработчик созданной заготовки объекта.

Create

Создает заготовку объекта на основе прототипа с идентификатором prototype. Если коллекция типизированная, то создается заготовка того же типа, что и коллекция, а иначе тип берется у прототипа.

Синтаксис:

```
IDBObject Create(Int64 prototypeID);
```

Параметры:

prototypeID

Идентификатор версии объекта-прототипа.

Результат:

Обработчик созданной заготовки объекта.

CreateVersion

Создает заготовку новой версии объекта на основе версии объекта с идентификатором objectID.

Синтаксис:

```
IDBObject CreateVersion(Int64 objectID);
```

Параметры:

objectID

Идентификатор версии объекта.

Результат:

Обработчик созданной заготовки объекта.

DeleteAttribute

Удаляет атрибут attributeID у версий объектов objectIDs (если у объекта есть этот атрибут). Если значение параметра ignoreExceptions равно **true**, то ошибки удаления атрибута игнорируются и происходит удаление атрибута у следующего объекта в списке, если это возможно.

Синтаксис:

```
CommandResult DeleteAttribute(Int64[] objectIDs, object attributeID, bool ignoreExceptions);
```

Параметры:

objectIDs

Массив идентификаторов версий объектов.

attributeID

Идентификатор удаляемого атрибута.

ignoreExceptions

Определяет, нужно ли генерировать исключение при ошибке удаления атрибута.

Результат:

Структура с результатами выполнения метода.

EditAttribute

Изменяет значение атрибута attributeID у версий объектов objectIDs на значение values (если значение values равно **null**, то очищает значение). Если значение ignoreExceptions равно **true**, то ошибки изменения значений игнорируются и изменяется значение атрибута у следующего объекта в списке, если это возможно.

Синтаксис:

```
CommandResult EditAttribute (Int64[] objectIDs, object attributeID, object[] values, bool ignoreExceptions);
```

Параметры:

objectIDs

Массив идентификаторов версий объектов.

attributeID

Идентификатор удаляемого атрибута.

values

Значения атрибутов.

ignoreExceptions

Определяет, нужно ли генерировать исключение при ошибке изменения значений атрибута.

Результат:

Структура с результатами выполнения метода.

GetObjectVersionFiltrationState

Метод получает статус версии объекта согласно указанному правилу подбора версий.

Синтаксис:

```
ObjectFiltrationState GetObjectVersionFiltrationState (Int64 objectID, VersionsRule rule);
```

Параметры:

objectID

Идентификатор проверяемой версии объекта.

rule

Правило подбора версий.

Результат:

Статус версии объекта согласно указанному правилу подбора версий.

5.5.2 Интерфейс IDBObjectService

Серверный интерфейс, позволяющий получать обработчики объектов.

5.5.2.1 Пространство имён

Intermech.Interfaces

5.5.2.2 Способы получения интерфейса

Интерфейс доступен на серверной стороне. Получить его можно у статического контейнера сервисов **Intermech.Interfaces.Server.ServerServices**:

```
ServerServices.GetService(typeof(IDBObjectService)) as IDBObjectService.m
```

5.5.2.3 Методы интерфейса

GetObject

Метод позволяет получить обработчик для объекта с указанным идентификатором. При необходимости можно получить рабочую копию объекта, если она доступна.

Синтаксис:

```
IDBObject GetObject (IUserSession uSession, Int64 objectID, bool failIfNotFound, bool getWorkCopy);
```

Параметры:

uSession

Сессия, в рамках которой выполняется работа.

objectID

Идентификатор искомого объекта.

failIfNotFound

Если указать значение **true**, то ядро сгенерирует исключение, если указанный объект не будет найден.

getWorkCopy

Если указать значение **true**, то ядро вернёт рабочую копию объекта, если она есть у пользователя указанной сессии.

Результат:

Обработчик искомого объекта или его рабочей копии.

Данный интерфейс используется пользовательской сессией `UserSession` в методах `GetObject`.

5.5.3 Интерфейс ICreatorContainer

Интерфейс для управления коллекцией, создающей обработчиков объектов/связей указанных типов.

5.5.3.1 Пространство имён

Intermech.Interfaces.Server

5.5.3.2 Способы получения интерфейса

Интерфейс доступен на серверной стороне. Получить его можно у статического контейнера сервисов **Intermech.Interfaces.Server.ServerServices**:

`ServerServices.GetService(typeof(IDBObjectService)) as ICreatorContainer;`

Класс, который реализует интерфейс **IDBObjectService**, реализует и интерфейс **ICreatorContainer**.

5.5.3.3 Методы интерфейса

AddCreator

Метод добавляет в коллекцию создатель объектов, связей и т.п. определённого типа. При необходимости перекрывает ранее добавленный создатель.

Синтаксис:

`void AddCreator (object creatorType, object creatorInstance);`

`void AddCreator (object creatorType, object creatorInstance, bool replace);`

Параметры:

`creatorType`

Тип создаваемых объектов, атрибутов, т.п. Обычно передаётся Guid типа.

`creatorInstance`

Создатель обработчиков объектов, атрибутов и т.п. указанного типа (например, интерфейс **IDBObjectCreator**, создающий обработчики объектов).

`replace`

Если указать значение **true**, то будет выполнена замена ранее зарегистрированного создателя (если он есть в коллекции).

5.5.4 Интерфейс IDBObjectCreator

Интерфейс для создания новых объектов.

5.5.4.1 Пространство имён

Intermech.Interfaces.Server

5.5.4.2 Методы интерфейса

CreateObject

Метод позволяет создать обработчик экземпляра версии объекта определенного типа.

Синтаксис:

`IDBObject CreateObject (IUserSession uSession, Guid guid, DataTable objectParams);`

Параметры:

`uSession`

Сессия, в рамках которой выполняется работа.

`guid`

Глобальный идентификатор типа создаваемого объекта.

`objectParams`

Запись из таблицы **IMS_OBJECTS** с обязательными атрибутами версии объекта, обработчик которой нужно создать.

Результат:

Обработчик созданного экземпляра версии объекта.

5.5.5 Интерфейс IDBObjectCollectionCreator

Интерфейс создателей объектов типа **IDBObjectCollection**. Позволяет создавать собственные обработчики коллекции объектов определенного типа.

5.5.5.1 Пространство имён

Intermech.Interfaces.Server

5.5.5.2 Методы интерфейса

CreateObjectCollection

Создать экземпляр получателя списка объектов типа `objectTypeID`.

Синтаксис:

```
IDBObjectCollection CreateObjectCollection(IUserSession uSession, Guid guid, int objectTypeID);
```

Параметры:

`uSession`

Сессия пользователя.

`guid`

Глобальный идентификатор типа объектов.

`objectTypeID`

Идентификатор типа объектов.

5.6 Примеры

5.6.1 Как создавать объекты базы данных

1. Получаем коллекцию объектов.
2. `IDBObjectCollection objects = UserSession.GetObjectCollection(тип_объекта);`
3. Создаем заготовку объекта. Заготовка автоматически инициализируется атрибутами, соответствующими типу объекта. В `Create` также можно подать объект-прототип.
4. `IDBObject obj = objects.Create();`
5. Заполняем необходимые атрибуты и устанавливаем связи заготовки объекта.
6. Переводим заготовку в объект методом `CommitCreation(bool)`. Параметр этого метода позволяет указать необходимость удаления заготовки в случае возникновения исключения в процессе выполнения метода.

`obj.CommitCreation(true);` // или `obj.Delete(0)` если нужно отменить создание объекта.

5.6.2 Как получить список объектов

1. Получаем интерфейс коллекции объектов.

```
IDBObjectCollection objects = UserSession.GetObjectCollection(-1);
```

Параметр в `GetObjectCollection` означает тип объектов, с которыми мы хотим работать. Если `-1`, то работа будет идти с объектами всех типов. Следует отметить, что получение интерфейса `IDBObjectCollection` просто создает объект-обработчик коллекций объектов и не приводит к получению самих объектов из базы данных.

2. Формируем управляющую структуру для получения списка объектов.

```
using Intermech.Kernel.Search;
```

...

Создаем список условий, которым должны отвечать получаемые объекты. В данном примере мы получаем список объектов, взятых на изменение текущим пользователем, у которых атрибут “Наименование” начинается со строки **Комбайн**:

```
ConditionStructure[] conditions = new ConditionStructure[] {  
    new ConditionStructure((int)ObligatoryObjectAttributes.F_CHKOUT_BY, RelationalOperators.Equal,  
        UserSession.UserID, LogicalOperators.AND, 0, false),  
    new ConditionStructure(“Наименование”, RelationalOperators.StartString, “Комбайн”,  
        LogicalOperators.NONE, 0, true)};
```

Создаем список колонок, которые хотим получить в таблице со списком объектов. Для нетипизированного списка объектов необходимо обязательно указывать список атрибутов, которые сервер должен включить в

результатирующую таблицу. Для списка объектов определенного типа можно отправить `columns = null` – в этом случае сервер сформирует список колонок в соответствии с настройками атрибутов для данного типа объектов. Однако рекомендуется всегда указывать набор колонок для того, чтобы сервер вернул только необходимые данные. Также в целях оптимизации выполнения запроса настоятельно рекомендуется использовать коллекции объектов с указанием типа обрабатываемых объектов.

Существует 2 варианта заполнения списка колонок – через массив `ColumnDescriptor[]` или через раздельное заполнение списков колонок, сортировок и дополнительной информации по колонкам. В первом случае для описания каждой колонки нужно создать структуру `ColumnDescriptor`, в которой указать следующую информацию:

- 1) – идентификатор атрибута (числовой, глобальный или наименование),
- 2) – принадлежность атрибута (если указана **Auto**, то принадлежность определяется типом коллекции, у которой вызвали метод `Select` – для `IDBObjectCollection` по умолчанию берутся атрибуты объектов),
- 3) – вид информации, которую нужно получить из атрибута (текстовую, идентификационную и т.п.),
- 4) – способ именования колонок в таблице, получаемой оператором `Select`,
- 5) – порядок сортировки данных, содержащихся в колонке (`SortOrders.ASC` – по возрастанию, `SortOrders.DESC` – по убыванию, `SortOrders.NONE` – не сортировать),
- 6) – приоритет сортировки по данной колонке (сначала сортировка идет по колонке с приоритетом 1, затем – 2 и т.д.).

В данном примере таблица будет содержать идентификатор объектов, заголовок объектов, наименование и обозначение объектов. Данные будут отсортированы по атрибуту **Наименование** в порядке возрастания, а по идентификатору объектов – в порядке убывания.

```
ColumnDescriptor[] descriptors = new ColumnDescriptor[]{
    new ColumnDescriptor (ObligatoryObjectAttributes.F_OBJECT_ID, AttributeSourceTypes.Auto,
        ColumnContents.Text, ColumnNameMapping.Default, SortOrders.DESC, 2),
    new ColumnDescriptor (ObligatoryObjectAttributes.CAPTION, AttributeSourceTypes.Auto,
        ColumnContents.Text, ColumnNameMapping.Default, SortOrders.NONE, 0),
    new ColumnDescriptor ("Наименование", AttributeSourceTypes.Auto, ColumnContents.Text,
        ColumnNameMapping.Default, SortOrders.ASC, 1),
    new ColumnDescriptor (new Guid("cad0001f-306c-11d8-b4e9-00304f19f545"))
}
```

Аналогичный запрос можно сформировать посредством массивов с идентификаторами колонок и сортировки:

```
object[] columns = new object[]{ObligatoryObjectAttributes.F_OBJECT_ID,
    ObligatoryObjectAttributes.CAPTION, "Наименование", new Guid("cad0001f-306c-11d8-b4e9-00304f19f545")};
```

Далее укажем порядок сортировки объектов в таблице. При этом нужно учитывать следующее: 1) в `sortColumns` можно указывать только те атрибуты, которые присутствуют в `columns`, 2) количество элементов в `sortColumns` и `orders` должно совпадать.

```
object[] sortColumns = new object[]
{"Наименование", ObligatoryObjectAttributes.F_OBJECT_ID};
SortOrders[] orders = new SortOrders[]{SortOrders.ASC, SortOrders.DESC};
```

Формируем структуру с параметрами запроса к серверу. Параметр `QueryConsts.Default` означает, что сервер должен вернуть пакет с количеством записей, определенных администратором системы по умолчанию. Далее идут параметры `lastKeyValue` и `lastOrderValue`, которые должны содержать значения ключевого поля и поля сортировки для последней записи, содержащейся в предыдущем пакете данных. Для получения первого пакета, а также при получении всего набора данных, `lastKeyValue=0` и `lastOrderValue=null`. Параметр **MyObjects** содержит `TableName` возвращаемой таблицы.

```
DBRecordSetParams pars = new DBRecordSetParams(cond, columns, sortColumns, orders, 0, null,
    QueryConsts.Default, true, "MyObjects");
```

3. Получаем таблицу со списком запрошенных объектов. Если в `GetObjectCollection` был передан тип объектов, то в список попадут только объекты этого и всех дочерних типов.

```
DataTable tbl = objects.Select(pars);
```

Возвращаемая таблица содержит параметр `tbl.ExtendedProperties["Eof"]`. Если `Convert(tbl.ExtendedProperties["Eof"]) == true`, то запрос вернул не все записи, отвечающие заданным условиям.

5.6.3 Как создавать свои обработчики объектов определённого типа

1. Создаём новый интерфейс обработчика объектов, который будет доступен на серверной и на клиентской сторонах. В данном интерфейсе объявим свойства и методы, которые потребуются от нового типа обработчика объектов.
2. Создаём новый класс обработчика объектов, унаследованный от базового класса `Intermech.Kernel.DBObject`. В данном классе реализуем необходимый функционал, перекрываем необходимые базовые методы, т.п., а также реализуем созданный интерфейс.
3. Создаём новый класс по созданию обработчиков, унаследованный от базового класса `Intermech.Kernel.DBObjectCreator`. В данном классе перекрываем метод `CreateObject`:

```
public override IDBObject CreateObject(IUserSession uSession, Guid guid, DataTable objectParams)
{
    return new новый_класс(uSession as UserSession, objectParams);
}
```

В данном методе возвращается экземпляр класса с новым обработчиком объектов (пункт 2).

4. Получаем на сервере приложений ссылку на интерфейс `ICreatorContainer` (он реализован в классе, реализующем интерфейс `IDBObjectService`):

```
ICreatorContainer objectService = ServerServices.GetService(typeof(IDBObjectService)) as
ICreatorContainer;
```

5. Регистрируем в данном интерфейсе класс по созданию обработчиков (пункт 3):

```
objectService.AddCreator(Guid типа объекта, new создатель_обработчиков());
```

После выполнения данных действий методы, возвращающие ссылку на интерфейс обработчика объектов указанного в пункте 5 типа объекта, будут возвращать ссылку на интерфейс `IDBObject`, который можно приводить к интерфейсу из пункта 1. Соответствующие примеры на языке C# входят в комплект поставки IPS Development Kit.

5.6.4 Как подписаться на изменение значения атрибута объектов и связей

1. Пишем обработчик события записи нового значения атрибута. С помощью аргумента `args` можно получить доступ к старому `args.OldValue` и новому `args.Value` значению атрибута, а также изменить его значение параметром `args.NewValue`.

```
private void WriteAttributeValue(IDBAttribute attribute, AttributeValueEventArgs args)
{
    //Тут выполняем какие-либо действия – в данном случае приводим новое значение к верхнему
    //регистру. А можно что-либо проверить и сгенерировать исключение.
    args.NewValue = args.Value.ToString().ToUpper();
    //А через args.Session можно получить доступ к сессии, от имени которой производится
    //модификация атрибута
}
```

2. В процедуре загрузки Вашего модуля привязываем обработчик на запись значения в атрибут номер `myAttributeID`. В качестве номера атрибута в `AddAttributeWriteHandler` можно отправить наименование или `Guid` атрибута.

```
public void Load(IServiceProvider serviceProvider)
{
    ...
    IEventLogHelper _EventLogHelper = ServerServices.GetService(typeof(IEventLogHelper)) as IEventLogHelper;
```

```

_EventLogHelper.AddAttributeWriteHandler(myAttributeID, new
WriteAttributeValueHandler(WriteAttributeValue));
...
}

```

5.6.5 Создание обработчиков объектов своих типов

1. Пишем класс-обработчик собственного объекта, порождая его от `Intermech.Kernel.DBObject` и реализуя в нем (при необходимости) свои дополнительные интерфейсы.

```

class MyDBObject: DBObject, IDBMyInterface
{
    override ObjectMethod() // Перекрываем имеющиеся методы
    {
    }
    public MyMethod() // И реализуем методы своих интерфейсов
    {
    }
}

```

2. Выбираем Guid `MyGuid` для своего типа объектов и создаем этот тип в базе данных.
3. Пишем создатель объектов данного типа и реализуем в нем поддержку интерфейса `Intermech.Interfaces.IDBObjectCreator`.

```

class MyObjectCreator: IDBObjectCreator
{
    public IDBObject CreateObject(IUserSession uSession, Guid guid, DataTable objectParams)
    {
        if (guid == MyGuid) // Это если Ваш создатель поддерживает много типов объектов (а не только
//MyGuid). Но можно писать свой отдельный создатель для каждого типа объектов.
        {
            return new MyDBObject(uSession, objectParams);
        }
    }
}

```

4. При загрузке своего плагина регистрируем свой создатель объектов для типа `MyGuid`.

```

...
IDBObjectCreator creator = new MyObjectCreator();
ICreatorContainer creators = ServerServices.GetService(typeof(IDBObjectService)) as ICreatorContainer;
creators.AddCreator(MyGuid, creator);
...

```

6.1 Связь между объектами

Связь – это категория, устанавливающая подчиненное отношение между версией информационного объекта и другим информационным объектом, т.е. в версию объекта с идентификатором ObjectID входит объект с идентификатором ID. Такой способ связывания позволяет оставлять без изменения связи родительского объекта при выпуске новой версии любого из дочерних объектов. Конкретизация версий дочерних объектов при получении состава родительского объекта производится с помощью текущего правила подбора версий. Все связи делятся на типы, которые определяют набор допустимых атрибутов у связи, видимость связей в клиентских приложениях, хранение истории связей и т.д.

6.1.1 Интерфейс IDBRelation

Интерфейс связи между объектами. Реализация интерфейса осуществляется объектом **IDBRelation**. Интерфейс унаследован от интерфейсов **IDBAttributable**, **IDBSessionable**.

6.1.1.1 Пространство имён

Intermech.Interfaces

6.1.1.2 Способы получения интерфейса

Для получения интерфейса используются функции:

```
IUserSession. GetRelation(Int64 aRelationID);  
IUserSession. GetRelation(Guid guid);  
IUserSession. GetRelation(Int64 projectID, Int64 partID, int relationType, bool versionMode);  
IUserSession. GetRelation(Int64 projectID, Int64 partID, bool versionMode);  
IUserSession. GetRelation(Int64 projectID, Int64 partID, int relationType);  
IUserSession. GetRelation(Int64 projectID, Int64 partID);
```

Параметры:

aRelationID

Идентификатор связи.

guid

Глобальный идентификатор связи.

projectID

Идентификатор версии родительского объекта.

partID

Идентификатор дочернего объекта.

relationType

Тип связи.

versionMode

Если значение versionMode равно **true**, то partID это ObjectID, иначе ID дочернего объекта.

6.1.1.3 Свойства интерфейса

CreateDate

Дата создания связи. Значение всегда округляется до даты. Если значение равно DateTime.MinValue, то это значит, что там пустое значение.

Синтаксис:

```
DateTime CreateDate { get; set; }
```

DeleteDate

Данное поле более не используется в системе.

Синтаксис:

```
DateTime DeleteDate { get; }
```

FiltrationOwnerID

Уникальный ключ, по которому сервер определяет настройки фильтрации состава.

Синтаксис:

```
string FiltrationOwnerID { get; set; }
```

GUID

Глобальный идентификатор связи.

Синтаксис:

```
Guid GUID { get; set; }
```

PartID

Идентификатор дочернего объекта (IDBObject.ID).

Синтаксис:

```
Int64 PartID { get; }
```

PartObjectID

Идентификатор версии дочернего объекта (IDBObject.ObjectID, например, если связь рассматривается в контексте конкретной версии объекта). Если идентификатор версии получить не удалось, то значение равно 0.

Синтаксис:

```
Int64 PartObjectID { get; }
```

ProjID

Идентификатор версии объекта-родителя (IDBObject.ObjectID).

Синтаксис:

```
Int64 ProjID { set; get; }
```

RelationID

Идентификатор связи (только для чтения). Следует учитывать, что идентификатор связи изменяется у рабочей копии объекта, поэтому после операций CheckOut и CheckIn связи будут иметь другие идентификаторы. Только глобальный идентификатор связи остается неизменным на протяжении всего жизненного цикла связи.

Синтаксис:

```
Int64 RelationID { get; }
```

RelationType

Идентификатор типа связи.

Синтаксис:

```
int RelationType { get; set; }
```

6.1.1.4 Методы интерфейса

Delete

Удалить связь.

Синтаксис:

```
int Delete(Int64 DeleteMode);
```

Параметры:

DeleteMode

Параметр зарезервирован для будущего использования.

Возвращаемое значение:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

6.1.2 Класс DBRelation

Стандартный обработчик связей между объектами. Класс унаследован от DBAttributable, реализует интерфейсы IDBRelation, IDeletable, IDBGuid, IDBAttributable, IDBSessionable. При написании собственных обработчиков связей определенного типа следует наследоваться от данного класса.

6.1.2.1 Пространство имён

Intermech.Kernel

6.1.2.2 Свойства класса

Applicability

Ссылка на применяемость - интерфейс, описывающий контекст связи типа RelationType между объектами типа ObjectType (дочерний) и InObjectType (родительский).

Синтаксис:

```
public IDBRelationsApplicability Applicability { get; }
```

Attributes

Коллекция атрибутов связи.

Синтаксис:

```
public override IDBAttributeCollection Attributes { get; }
```

ObjectName

Название связи.

Синтаксис:

```
public override string ObjectName { get; }
```

ReadOnly

Допускается ли изменение связи.

Синтаксис:

```
public bool ReadOnly { get; }
```

RelationTypeObject

Ссылка на интерфейс, описывающий тип данной связи.

Синтаксис:

```
public IDBRelationType RelationTypeObject { get; }
```

6.1.2.3 Методы класса

DoDelete

Выполняет действия по удалению связи. Данный метод можно переопределить в собственном обработчике связей для выполнения дополнительных проверок и действий, связанных с удалением связи. Вызов базовой реализации метода обязателен!

Синтаксис:

```
protected virtual int DoDelete(Int64 DeleteMode);
```

Параметры:

DeleteMode

Содержит флаги, управляющие процессом удаления связи.

Результат:

Всегда возвращает 0. В случае каких-либо ошибок удаления метод генерирует исключения.

DoRemove

Выполняет действия по перемещению связи в состав другого объекта. Данный метод можно переопределить в собственном обработчике связей для выполнения дополнительных проверок и действий, связанных с перемещением связи. Вызов базовой реализации метода обязателен!

Синтаксис:

```
protected virtual void DoRemove(Int64 newProjID, int newRelationTypeID);
```

Параметры:

newProjID

Идентификатор версии объекта, в состав которой нужно переместить данную связь.

newRelationTypeID

Тип, который нужно присвоить перемещаемой связи.

6.2 Создание и поиск связей

6.2.1 Интерфейс IDBRelationCollection

Интерфейс предназначен для работы со списками связей. Реализация осуществляется объектом DBRelationCollection. Интерфейс унаследован от IDBRecords, IDBSessionable.

6.2.1.1 Пространство имён

Intermech.Interfaces

6.2.1.2 Способы получения интерфейса

```
IUserSession.GetRelationCollection(int relationType);
```

```
IUserSession.GetRelationCollection(int relationType, string FiltrationOwnerID);
```

Параметры:

relationType

Тип связей, с которым должна работать коллекция. Если `relationType = -1`, то коллекция будет работать со связями всех типов. Чтобы ускорить получение списка связей рекомендуется указывать тип связей `relationType`.

FiltrationOwnerID

Уникальный идентификатор настроек фильтрации состава.

6.2.1.3 Свойства интерфейса

FiltrationOwnerID

Уникальный ключ, по которому сервер (сервис `IVersionRulesCacheService`) определяет настройки фильтрации состава. Вместо указания этого ключа можно явно задать правило подбора версий (поле `FiltrationRule`, приоритет которого выше, чем у поля `FiltrationOwnerID`).

Синтаксис:

```
string FiltrationOwnerID { get; set; }
```

FiltrationRule

Вместо указания ключа с настройками фильтрации (`FiltrationOwnerID`) можно явно задать правило подбора версий, по которому будет осуществляться фильтрация состава (приоритет поля `FiltrationRule` выше, чем у поля `FiltrationOwnerID`).

Синтаксис:

```
VersionsRule FiltrationRule { get; set; }
```

ObjectTypeID

Идентификатор типа объектов, которые нужно получить по списку связей (по умолчанию значение свойства равно `-1`, т.е. коллекция будет возвращать список объектов всех типов). Чтобы ускорить получение списка связей рекомендуется указывать также и тип связанных объектов.

Синтаксис:

```
int ObjectTypeID { get; set; }
```

RelationTypeID

Идентификатор типа связей, с которыми работает данная коллекция.

Синтаксис:

```
int RelationTypeID { get; set; }
```

6.2.1.4 Методы интерфейса

ActualDateObjectVersion

Метод позволяет отыскать версию объекта, которая существовала на указанную дату на указанном уровне продвижения или на указанном шаге ЖЦ (значения даты и уровня/шага ЖЦ находятся в правиле подбора версий). Если версия не найдена, вернётся значение `-1`.

Синтаксис:

```
Int64 ActualDateObjectVersion (Int64 objectID, VersionsRule rule, out ObjectFiltrationState state);
```

Параметры:

`objectID`

Идентификатор версии объекта.

`rule`

Специальным образом подготовленное правило подбора версий. В правиле должна быть указана актуальная дата состава, а в критериях правила должны быть уровень продвижения объекта или шаг жизненного цикла.

`state`

В данном параметре будет возвращён статус найденной версии объекта.

Результат:

Найденный идентификатор версии объекта или `-1`.

ConsistFrom

Получить состав версии объекта с идентификатором `projectID` на текущую дату в соответствии с условиями структуры `paramSet`.

Синтаксис:

```
DataTable ConsistFrom(DBRecordSetParams paramSet, Int64 projectID);
```

```
DataTable ConsistFrom(DBRecordSetParams paramSet, Int64 projectID, bool recursive);
```

```
DataTable ConsistFrom(DBRecordSetParams paramSet, Int64 projectID, bool recursive, DateTime actualDate);
```

Параметры:

paramSet

Структура свойств, описывающих выборку записей из базы данных, которую нужно передать на клиента, включающая условия поиска и информацию о колонках выборки.

projectID

Идентификатор версии объекта.

recursive

Если значение параметра recursive равно true, то метод получает развернутый состав со всем деревом связей.

actualDate

Локальная дата, на которую нужно получить связи.

Результат:

Состав версии объекта.

ConsistFromBlanks

Получает массив идентификаторов объектов-заготовок, которые в данный момент входят в состав версии объекта projectID текущим типом связи.

Синтаксис:

```
Int64[] ConsistFromBlanks(Int64 projectID);
```

Параметры:

projectID

Идентификатор версии родительского объекта.

Результат:

Массив идентификаторов объектов-заготовок (IDBObject.ObjectID).

Create

Создает связь между версиями родительского и дочернего объектов. Тип связи задается при получении объекта IDBRelationCollection (идентификатор типа связи должен быть неотрицательным). Связь начинает действовать либо с момента ее создания, либо с даты beginDate.

Синтаксис:

```
IDBRelation Create(Int64 projectID, Int64 partObjectID);
```

```
IDBRelation Create(Int64 projectID, Int64 partObjectID, DateTime beginDate);
```

```
IDBRelation Create(Int64 projectID, Int64 partObjectID, DateTime beginDate, DateTime endDate);
```

```
IDBRelation Create(NewRelationProperties properties);
```

Параметры:

projectID

Идентификатор версии родительского объекта.

partObjectID

Идентификатор версии дочернего объекта.

beginDate

Локальная дата начала действия связи.

endDate

Локальная дата окончания действия связи.

properties

Структура с параметрами для создания новой связи. С помощью данной структуры можно сразу присвоить создаваемой связи значения дополнительных атрибутов, указать в качестве прототипа интерфейс существующей связи, задать глобальный идентификатор создаваемой связи и пр.

Результат:

Интерфейс обработчика созданной связи.

DeleteRelations

Удаляет набор дочерних связей у заданной версии объекта.

Синтаксис:

```
int DeleteRelations(Int64 projID, IList<Guid> relationsGUID, bool transactionMode,
```

```
Int64 deleteMode);
```

Параметры:

projID

Идентификатор версии родительского объекта.

relationsGUID

Список уникальных глобальных идентификаторов удаляемых связей.

transactionMode

Если содержит **true**, то удаление производится в одной транзакции, т.е. в случае ошибки будут отменены все удаления.

deleteMode

Зарезервировано. Указывайте значение **0**.

Результат:

Количество удалённых связей.

EntersIn

Получить применяемость объекта с идентификатором partID на текущую дату в соответствии с условиями структуры paramSet.

Синтаксис:

```
DataTable EntersIn(DBRecordSetParams paramSet, Int64 partID);
```

```
DataTable EntersIn(DBRecordSetParams paramSet, Int64 partID, bool recursive, DateTime actualDate);
```

Параметры:

paramSet

Структура свойств, описывающих выборку записей из базы данных, которую нужно передать на клиента.

partID

Идентификатор дочернего объекта (IDBObject.ID), применяемость которого нужно получить. Если значение параметра равно -1, то по этому параметру связи не фильтруются.

actualDate

Локальная дата, на которую нужно получить связи.

recursive

Если значение параметра равно *true*, то метод получает развернутый список применяемости во всем дереве связей.

Результат:

Применяемость объекта.

EntersInVersion

Получить применяемость версии объекта номер partID (IDBObject.ObjectID) на текущую дату в соответствии с условиями paramSet.

Синтаксис:

```
DataTable EntersInVersion(DBRecordSetParams paramSet, Int64 objectID);
```

```
DataTable EntersInVersion(DBRecordSetParams paramSet, Int64 objectID, bool recursive, DateTime actualDate);
```

Параметры:

paramSet

Структура свойств, описывающих выборку записей из базы данных, которую нужно передать на клиента, включающая условия поиска и информацию о колонках выборки.

objectID

Идентификатор версии объекта, применяемость которого нужно получить.

recursive

Если значение параметра равно *true*, то метод получает развернутый список применяемости во всем дереве связей.

actualDate

Локальная дата, на которую нужно получить связи.

Результат:

Применяемость версии объекта.

GetObjectVersionFiltrationState

Метод позволяет получить статус указанной версии объекта согласно указанному правилу подбора версий.

Синтаксис:

```
ObjectFiltrationState GetObjectVersionFiltrationState (Int64 objectID, VersionsRule rule);
```

Параметры:

objectID

Идентификатор версии объекта.

rule

Правило подбора версий.

Результат:

Статус версии объекта согласно указанному правилу подбора версий.

IsObjectInFolders

Возвращает значение *true*, если объект с идентификатором *id* (IDBObject.ID) входит в любой из объектов *parentsObjectID* (IDBObject.ObjectID). Тип связи берётся из текущей коллекции связей.

Синтаксис:

```
bool IsObjectInFolders(Int64 id, Int64[] parentsObjectID);
```

Параметры:

id

Идентификатор объекта.

parentsObjectID

Массив идентификаторов версий объектов.

Результат:

Результаты проверки.

Select

Получить таблицу со списком связей, соответствующую *paramSet*. Тип связей указывается при получении объекта, реализующего *IDBRelationCollection* (если при получении указано значение типа меньше 0, то в запрос включаются связи всех типов).

Синтаксис:

```
DataTable Select(DBRecordSetParams paramSet, Int64 projectID, Int64 partID, DateTime actualDate);
```

Параметры:

paramSet

Структура свойств, описывающих выборку записей из базы данных, которую нужно передать на клиента.

projectID

Идентификатор версии родительского объекта, чей состав нужно получить. Если значение параметра равно -1, то по этому параметру связи не фильтруются.

partID

Идентификатор дочернего объекта (IDBObject.ID), применяемость которого нужно получить. Если значение параметра равно -1, то по этому параметру связи не фильтруются.

actualDate

Локальная дата, на которую нужно получить связи.

Результат:

Таблица связей между объектами. Набор колонок указывается в параметре *paramSet*. Ниже приведена таблица с обязательными атрибутами связей.

Таблица IMS_RELATIONS. Список связей.

Имя поля	Тип поля	Комментарий	IDBRelation
F_PRJLINK_ID	INTEGER	Идентификатор связи.	RelationID
F_PROJ_ID	INTEGER	Идентификатор версии родительского объекта.	ProjectID
F_PART_ID	INTEGER	Идентификатор дочернего объекта.	PartID
F_RELATION_TYPE	INTEGER	Идентификатор типа связи.	RelationType
F_CREATE_DATE	DATE	Дата создания связи.	CreateDate
F_DELETE_DATE	DATE	Данное поле не используется в системе.	
F_PRJ_GUID	VARCHAR(40)	Глобальный идентификатор связи.	IDBGuid.GUID

6.2.2 Интерфейс IDBRelationService

Серверный интерфейс, позволяющий получать обработчики связей.

6.2.2.1 Пространство имён

Intermech.Interfaces.Server

6.2.2.2 Методы интерфейса

GetRelation

Метод позволяет получить обработчик для связи с указанным идентификатором, либо по идентификаторам дочернего и родительского объектов.

Синтаксис:

```
IDBRelation GetRelation(IUserSession uSession, Guid guid, Int64 projectID);
IDBRelation GetRelation(IUserSession uSession, Guid guid, Int64 projectID, bool failIfNotFound);
IDBRelation GetRelation(IUserSession uSession, Int64 projectID, Int64 partID, Int32 relationType);
IDBRelation GetRelation (IUserSession uSession, Int64 relationID, bool failIfNotFound);
```

Параметры:

uSession

Сессия, в рамках которой выполняется работа.

guid

Уникальный глобальный идентификатор связи.

relationID

Идентификатор связи.

failIfNotFound

Если указать значение **true**, то ядро сгенерирует исключение, если связь не была найдена. Иначе при отсутствии связи функция вернёт **null**.

projectID

Идентификатор версии родительского объекта (IDBObject.ObjectID).

partID

Идентификатор дочернего объекта (IDBObject.ID).

relationType

Идентификатор типа связи.

Результат:

Обработчик связи или **null**.

6.2.3 Интерфейс IDBRelationCreator

Интерфейс объекта, создающего обработчики связей указанного типа. Используется для создания собственных обработчиков связей.

6.2.3.1 Пространство имён

Intermech.Interfaces.Server

6.2.3.2 Методы интерфейса

CreateRelation

Создает обработчик связей типа guid.

Синтаксис:

```
IDBRelation CreateRelation(IUserSession uSession, Guid guid, DataTable relationParams);
```

Параметры:

uSession

Сессия пользователя.

guid

Глобальный идентификатор типа связей.

relationParams

Обязательные параметры связи, полученные запросом SELECT * из таблицы связей между объектами IMS_RELATIONS.

Результат:

Обработчик связи.

6.2.4 Интерфейс IDBRelationCollectionCreator

Интерфейс создателей объектов типа DBRelationCollection.

6.2.4.1 Пространство имён

Intermech.Interfaces.Server

6.2.4.2 Методы интерфейса

CreateRelationCollection

Создать экземпляр получателя списка связей типа relationTypeID.

Синтаксис:

```
IDBRelationCollection CreateRelationCollection(IUserSession uSession, Guid guid, int relationTypeID);
```

Параметры:

uSession

Сессия пользователя.

guid

Глобальный идентификатор типа связей.

relationTypeID

Идентификатор типа связей.

Результат:

Коллекция связей.

6.2.5 Интерфейс ICompositionLoadService

Серверная служба, позволяющая читать составы по указанным правилам подбора версий, правилам сортировки и отображения составов.

6.2.5.1 Пространство имен

Intermech.Interfaces.Compositions

6.2.5.2 Методы интерфейса:

LoadComplexCompositions

Считать составы указанных родительских объектов по указанному типу связи (все составы загружаются одним запросом в одну общую таблицу).

Синтаксис:

```
System.Data.DataTable LoadComplexCompositions (object usrSession;
```

```
System.Collections.Generic.List<Int64> projIDs;
```

```
Int32 relationTypeID;
```

```
System.Collections.Generic.List<ColumnDescriptor> columns; Intermech.Interfaces.VersionsRule rule;
```

```
params Int32[] childObjectTypes)
```

```
System.Data.DataTable LoadComplexCompositions (object usrSession;
```

```
System.Collections.Generic.List<Int64> projIDs;
```

```
Int32 relationTypeID;
```

```
System.Collections.Generic.List<ColumnDescriptor> columns;
```

```
string filtrationOwnerID;
```

```
params Int32[] childObjectTypes)
```

Параметры:

usrSession

Пользовательская сессия. При обращении к сервису со стороны сервера сюда можно передавать ссылку на интерфейс IUserSession или GUID сессии (как строку или System.Guid). При обращении к кэшу со стороны клиента сюда можно передавать только GUID сессии (как строку или System.Guid).

projIDs

Список [F_OBJECT_ID] версий родительского объекта, для которых будет считываться состав.

relationTypeID

Тип связи, по которой надо читать составы.

columns

Коллекция столбцов для запроса состава из базы данных.

rule

Правило подбора версий, по которому будет фильтроваться состав.

childObjectTypes

Дополнительная типизация коллекции связей дочерними типами объектов.

Возвращаемое значение:

Datatable с указанными столбцами (если нет состава - вернёт пустую Datatable) или null в случае ошибки.

LoadComposition

Считать состав указанного родительского объекта.

Синтаксис:

```
System.Data.DataTable LoadCompositions (object usrSession;
Int64 projID;
System.Collections.Generic.List<ColumnDescriptor> columns;
Intermech.Interfaces.VersionsRule rule;
params Int32[] childObjectTypes)
System.Data.DataTable LoadCompositions (object usrSession;
Int64 projID;
System.Collections.Generic.List<ColumnDescriptor> columns;
string filtrationOwnerID;
params Int32[] childObjectTypes)
System.Data.DataTable LoadComposition (object usrSession;
Int64 projID;
Int32 relationTypeID;
System.Collections.Generic.List<ColumnDescriptor> columns;
Intermech.Interfaces.VersionsRule rule;
params Int32[] childObjectTypes)
System.Data.DataTable LoadComposition (object usrSession;
Int64 projID;
Int32 relationTypeID;
System.Collections.Generic.List<ColumnDescriptor> columns;
string filtrationOwnerID;
params Int32[] childObjectTypes)
System.Data.DataTable LoadCompositions (object usrSession; Int64
projID;
System.Collections.Generic.List< Int32> visibleRelTypes;
System.Collections.Generic.List<ColumnDescriptor> columns;
Intermech.Interfaces.VersionsRule rule;
params Int32[] childObjectTypes)
System.Data.DataTable LoadCompositions (object usrSession;
Int64 projID;
System.Collections.Generic.List< Int32> visibleRelTypes;
System.Collections.Generic.List<ColumnDescriptor> columns;
string filtrationOwnerID;
params Int32[] childObjectTypes)
```

Параметры:

usrSession

Пользовательская сессия. При обращении к сервису со стороны сервера сюда можно передавать ссылку на интерфейс IUserSession или GUID сессии (как строку или System.Guid). При обращении к кэшу со стороны клиента сюда можно передавать только GUID сессии (как строку или System.Guid).

projID

[F_ОБЪЕКТ_ID] версии родительского объекта, для которого будет считываться состав.

relationTypeID

Тип связи, по которому надо получить состав.

columns

Коллекция столбцов для запроса состава из базы данных.

rule

Правило подбора версий, по которому будет фильтроваться состав.

Возвращаемое значение:

Datatable с указанными столбцами (если нет состава - вернёт пустую Datatable) или null в случае ошибки.

filtrationOwnerID

Уникальный ключ настроек фильтрации состава. Если фильтрация состава не требуется, можно указать константу Intermech.SystemGUIDs.filtrationAllVersions.

visibleRelTypes

Список типов связей, по которым надо читать составы.

childObjectTypes

Дополнительная типизация коллекции связей дочерними типами объектов.

6.3 Фильтрация состава

При получении состава объекта для каждого входящего объекта в большинстве случаев необходимо подобрать одну версию (не больше и не меньше). Это делается с помощью правил подбора версий.

6.3.1 Правила подбора версий

Правила подбора версий представляют собой объекты базы данных определенного типа. От базового типа **Правило подбора версий** (SystemGUIDs.objtypeVersionRule) порождены два дочерних типа – **Общее правило подбора версий** (SystemGUIDs.objtypeVersionRuleCommon) и **Персональное правило подбора версий** (SystemGUIDs.objtypeVersionRuleUser). Они отличаются схемой жизненного цикла.

Кроме того, существуют системные правила, например **Все версии объектов**, **Последние версии объектов**. Эти правила нельзя изменять.

Для получения составов на какую-то конкретную дату в правиле подбора версий можно указать эту дату. В этом случае для корректного подбора версий состава в правиле также должны быть критерии с уровнем продвижения объектов и(или) с шагом жизненного цикла.

Для работы с правилами подбора версий используются экземпляры класса Intermech.Interfaces.VersionsRule.

6.3.1.1 Критерии подбора

Каждое правило подбора версий состоит из критериев подбора.

Критерий подбора включает в себя следующие компоненты:

- атрибут фильтруемой версии объекта;
- функция сравнения (и флажок отрицания её значения);
- значения для сравнения.

6.3.1.2 Функции сравнения, виды критериев подбора

Функции сравнения бывают двух видов - обычные и агрегатные. Обычная функция выполняет сравнение между значением атрибута и значениями для сравнения (либо проводит какие-либо вычисления над значением атрибута) и возвращает логическое «Истина» или «Ложь», что означает соответствие указанного атрибута значениям для сравнения. Агрегатная функция отыскивает минимальное или максимальное значение атрибута среди всех фильтруемых версий объектов.

Соответственно, по виду реализуемой функции сравнения, критерии подбора бывают двух видов - обычные и дополнительные. Если фильтрация всех версий объектов по обычным критериям подбора с помощью обычных функций сравнения не дала положительных результатов, или наоборот, дала несколько подходящих версий, то выполняется фильтрация по дополнительному критерию подбора по агрегатной функции сравнения и отыскивается единственная наиболее подходящая версия объекта. В связи с этим в каждом правиле подбора версий должен быть как минимум один обычный критерий подбора и только один дополнительный критерий подбора.

Список допустимых функций сравнения меняется в зависимости от типа данных атрибута для сравнения, а также от его вида (системный атрибут или пользовательский). Для атрибутов, представляющих собой простые типы данных (числовые, строковые), можно реализовать различные функции сравнения, но для сложных типов данных (например, BLOB), допустимо лишь проверять значение такого атрибута на NULL.

6.3.1.3 Аргументы функций сравнения - значения для сравнения

В зависимости от функции сравнения в качестве аргументов могут выступать не только атрибуты фильтруемой версии объекта, но и значения для сравнения.

Значения для сравнения бывают следующих видов:

- константа;
- переменная;
- другой атрибут фильтруемой версии объекта.

Значения-константы задаются при проектировании правила подбора версий. Значения-переменные получают какие-то начальные значения при проектировании правила подбора версий, а затем каждый

пользователь имеет возможность задавать несколько вариантов значений таких переменных. Перед началом фильтрации все переменные значения должны быть инициализированы. Значения-атрибуты заполняются непосредственно во время самой фильтрации на основании атрибутов фильтруемой версии объекта.

Критерии подбора объединяются между собой с помощью булевых функций **ИЛИ** и **И**. Кроме того, к результату вычисления значения критерия для сравнения может быть применено логическое отрицание **НЕ**.

6.3.1.4 Пример правила подбора версий

Рассмотрим небольшой пример. Правило состоит из трёх критериев подбора:

Пример правила подбора версий №1.

Атрибут	Отрицание	Функция сравнения, значения для сравнения	Операция
Номер извещения		РАВЕН Переменная	ИЛИ
Уровень продвижения		РАВЕН Константа "В производстве"	
Идентификатор версии		ИМЕЕТ МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ	

Первые два критерия являются основными. У них задана обычная функция для сравнения – **Равен**. Последний критерий основан на агрегатной функции **Имеет максимальное значение** и является дополнительным. Номер извещения (переменная) будет задан пользователем перед началом фильтрации.

6.3.1.5 Алгоритм фильтрации

Рассмотрим на вышеприведённом примере правила подбора №1, как будет выполняться фильтрация состава.

Сервер приложений возвращает таблицу версий объектов. Из таблицы выбирается очередная версия. У неё берётся значение её атрибута **Номер извещения** и сравнивается с указанным в правиле значением. В нашем случае это – переменная величина, которую задаёт пользователь, выбирая правило подбора перед началом фильтрации. Если значение атрибута совпадает со значением критерия, то фильтрация для данной версии объекта по правилу прекращается, версия попадает в список подходящих версий, а фильтрация продолжается для оставшихся версий объектов.

Если же номер извещения не подходит, то проводится дальнейшая фильтрация, уже по атрибуту **Уровень продвижения**. Алгоритм сравнения – тот же самый. При этом для каждой версии запоминается её «вес» – сколько основных критериев подбора потребовалось для того, чтобы версия прошла по ним фильтрацию.

В итоге, после фильтрации всех версий объекта по правилам получается два списка версий:

- версии, которые прошли фильтрацию по основным критериям подбора;
- версии, которые не прошли фильтрацию по основным критериям подбора.

Если в первом списке есть только одна версия, она сразу же признаётся полностью прошедшей фильтрацию и возвращается пользователю.

Если в первом списке присутствует несколько версий, или список подходящих версий пуст, но есть версии объектов во втором списке, то:

1. Список сортируется по возрастанию «весов» подобранных версий.
2. Из списка исключаются все версии, «вес» которых больше «веса» первой версии в списке (с минимальным «весом» - как наиболее подходящей по основным критериям).
3. Если в списке остаётся только одна версия, она возвращается со статусом «Единственная версия, прошедшая фильтрацию по основным критериям подбора».
4. Для всех версий в списке выполняется фильтрация по дополнительному критерию подбора среди этих версий объектов. Поочерёдно выбираются эти версии, среди них разыскивается версия объекта, у которого заданный атрибут (в примере – **Идентификатор версии**) имеет максимальное (или минимальное значение). Найденная версия возвращается пользователю.

Если такой поиск проводился среди версий, прошедших фильтрацию по основным критериям отбора, то версия помечается как «Полностью прошедшая фильтрацию, но отобранная по дополнительному критерию».

Если же поиск проводился среди версий из второго списка, то найденная версия помечается как «Версия, не совсем подходящая по критериям заданного правила подбора версий» («Найденная по дополнительному критерию среди неподходящих версий»).

6.3.1.6 Группирование критериев сравнения

Разработчик правила подбора версий может объединять несколько критериев подбора в группы с помощью логической операции **И**. В этом случае полностью вычисляются все значения этой объединённой группы критериев, затем между ними выполняется логическая операция **И** и результат рассматривается в целом.

Пример правила подбора версий №2.

Атрибут	Отрицание	Функция сравнения, значения для сравнения	Операция
Номер извещения		РАВЕН Переменная	ИЛИ
Номер извещения		РАВЕН Переменная	И
Уровень продвижения	НЕ	РАВЕН Константа "В производстве"	
Идентификатор версии		ИМЕЕТ МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ	

Сразу вычисляется первый критерий, а затем, если первый критерий не подошёл, то оба значения второго и третьего критериев рассматриваются как одно значение (группа критериев). И если второй и третий критерии вернут **Истина**, то данная версия успешно пройдёт фильтрацию.

6.3.1.7 Применимость функций сравнения к типам данных атрибутов

Все атрибуты имеют тип данных, заданный в перечислителе Intermech.FieldTypes.

Функции для сравнения реализованы с помощью класса Intermech.CompareFunctionsHelper.

Рассмотрим по порядку весь перечень доступных функций для сравнения, опишем их назначение и применимость к типам данных.

Предполагается, что А - значение атрибута в критерии подбора, CV_i - одно из значений для сравнения, если их несколько (i = 0..n), либо CV – если это единственное значение для сравнения. С этими значениями сравнивается величина А.

Функции сравнения.

Функция сравнения Имя функции сравнения	Назначение	Применима для типов данных FieldTypes
АТРИБУТ НЕ ПУСТ "NOTNULL"	Универсальная функция. Вычисляется значение выражения A ≠ NULL Если атрибут не найден, будет возвращено значение «Ложь»	Любые типы
ИМЕЕТ МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ "MAX"	Агрегатная функция. Используется для поиска максимума среди значений атрибута А у всех версий фильтруемого объекта. Для типа ftString регистр символов роли не играет.	ftString ftInteger ftDouble ftDateTime ftAutoInc

Функция сравнения Имя функции сравнения	Назначение	Применима для типов данных FieldTypes
ИМЕЕТ МИНИМАЛЬНОЕ ЗНАЧЕНИЕ “MIN“	Агрегатная функция. Используется для поиска минимума среди значений атрибута А у всех версий фильтруемого объекта. Для типа ftString регистр символов роли не играет.	ftString ftInteger ftDouble ftDateTime ftAutoInc
РАВЕН “EQUALS“	Обычная функция. Используется для сравнения значения двух значений: $A = CV$ («Истина» при их равенстве). Для типа ftString регистр символов роли не играет.	ftSystem ftBoolean ftString ftInteger ftDouble ftDateTime ftAutoInc
БОЛЬШЕ “GREATER“	Обычная функция. Вычисляется значение выражения $A > CV$ Для типа ftString регистр символов роли не играет.	ftSystem ftString ftInteger ftDouble ftDateTime ftAutoInc
БОЛЬШЕ ИЛИ РАВЕН “EQUALS_GREATER“	Обычная функция. Вычисляется значение выражения $A \geq CV$ Для типа ftString регистр символов роли не играет.	ftSystem ftString ftInteger ftDouble ftDateTime ftAutoInc
МЕНЬШЕ “LESS“	Обычная функция. Вычисляется значение выражения $A < CV$ Для типа ftString регистр символов роли не играет.	ftSystem ftString ftInteger ftDouble ftDateTime ftAutoInc
МЕНЬШЕ ИЛИ РАВЕН “EQUALS_LESS“	Обычная функция. Вычисляется значение выражения $A \leq CV$ Для типа ftString регистр символов роли не играет.	ftSystem ftString ftInteger ftDouble ftDateTime ftAutoInc
В СПИСКЕ ЗНАЧЕНИЙ “IN_LIST“	Обычная функция. Используется для сравнения значения двух значений: $A = CV_i$ («Истина» при их равенстве), где CV_i – множество значений для сравнения. Сравнение возвращает «Истина», если хотя бы для одного значения для сравнения выполняется указанное равенство. Для типа ftString регистр символов роли не играет.	ftSystem ftBoolean ftString ftInteger ftDouble ftDateTime ftAutoInc

Функция сравнения Имя функции сравнения	Назначение	Применима для типов данных FieldTypes
В УКАЗАННОМ ДИАПАЗОНЕ “IN_BOUNDS”	Обычная функция. Вычисляется значение выражения $(A > CV_1)$ И $(A < CV_2)$, где CV_1 – первый критерий сравнения, CV_2 – второй. Для типа ftString регистр символов роли не играет.	tSystem ftString ftInteger ftDouble ftDateTime ftAutoInc
В УКАЗАННОМ ДИАПАЗОНЕ (С ГРАНИЦАМИ) “IN_BOUNDS_INC”	Обычная функция. Вычисляется значение выражения $(A \geq CV_1)$ И $(A \leq CV_2)$, где CV_1 – первый критерий сравнения, CV_2 – второй. Для типа ftString регистр символов роли не играет.	tSystem ftString ftInteger ftDouble ftDateTime ftAutoInc
СОДЕРЖИТ ПОДСТРОКУ “CONTAINS”	Обычная функция. Проверяется наличие значения CV в атрибуте A (поиск наличия подстроки в строке). Регистр символов роли не играет.	ftString

Для функции **Атрибут не пуст** не требуются значения для сравнения. Её единственным аргументом является проверяемое значение атрибута версии объекта.

Для большинства функций используется фиксированное количество значений для сравнения (1 или 2).

Для функции «В списке значений» можно задавать от 1 до 100 значений для сравнения.

6.3.2 Варианты значений переменных в правилах подбора

Подготовленное к фильтрации правило подбора версий должно содержать корректно заполненные поля. Во всех критериях и значениях для сравнения должны быть заданы атрибуты и значения, все переменные значения должны быть корректно заполнены (не пустые). В противном случае правило считается недействительным, и фильтрация по нему проводиться не будет.

Рассмотрим типичную ситуацию. Есть правило подбора версий, в критериях которого используются переменные значения. В системе есть пользователи, каждый из которых хочет иметь возможность задавать свои варианты значений этих переменных и, выбирая один из вариантов, менять настройки фильтрации состава

Например, есть правило подбора версий с переменными значениями критерия подбора:

Пример правила подбора версий №3.

Атрибут	Отрицание	Функция сравнения, значения для сравнения	Операция
Дата создания объекта		В УКАЗАННОМ ДИАПАЗОНЕ (С ГРАНИЦАМИ) Переменная (01.07.2006) Переменная (31.07.2006)	
Идентификатор версии		ИМЕЕТ МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ	

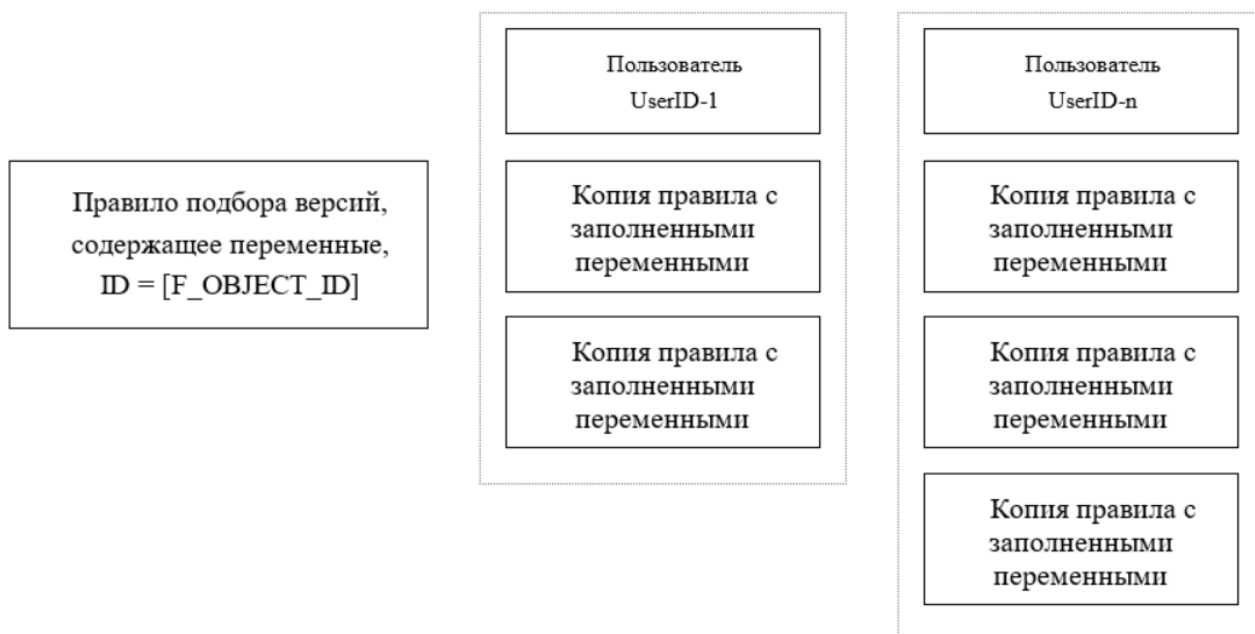
Первый пользователь может выполнять фильтрацию версий по нескольким своим диапазонам дат, второй пользователь – по своим диапазонам, и т.д.

Чтобы каждый раз не указывать значения переменных, существует механизм вариантов значений переменных в правилах подбора версий. Пользователь, выбрав параметризованное правило подбора, может создать для него несколько разных вариантов (фактически копий этого правила), указав в каждом из вариантов разные значения переменных. А перед фильтрацией он может быстро выбрать, по какому из вариантов указанного правила выполнять фильтрацию состава.

Указанные варианты значений для правил подбора хранятся в базе данных в настройках пользователя. Таким образом получается следующий механизм: есть параметризованное правило подбора версий (объект

базы данных), для него каждый пользователь может создавать варианты значений переменных этого правила. Варианты доступны только для пользователя, их создавшего.

В итоге получается следующий набор вариантов значений правил для нескольких пользователей:



Для получения корректного параметризованного правила подбора версий требуется использовать такой вариант «адресации»:

[UserID + F_ОБЪЕКТ_ID правила + индекс его варианта значений]

На практике такая адресация (с точки зрения программиста) упрощена. Для этих целей реализован интерфейс `Intermech.Interfaces.IFiltrationSettings`, который по уникальному строковому ключу позволяет получать любые варианты параметризованных правил подбора версий для указанного пользователя.

6.3.3 Настройки фильтрации составов.

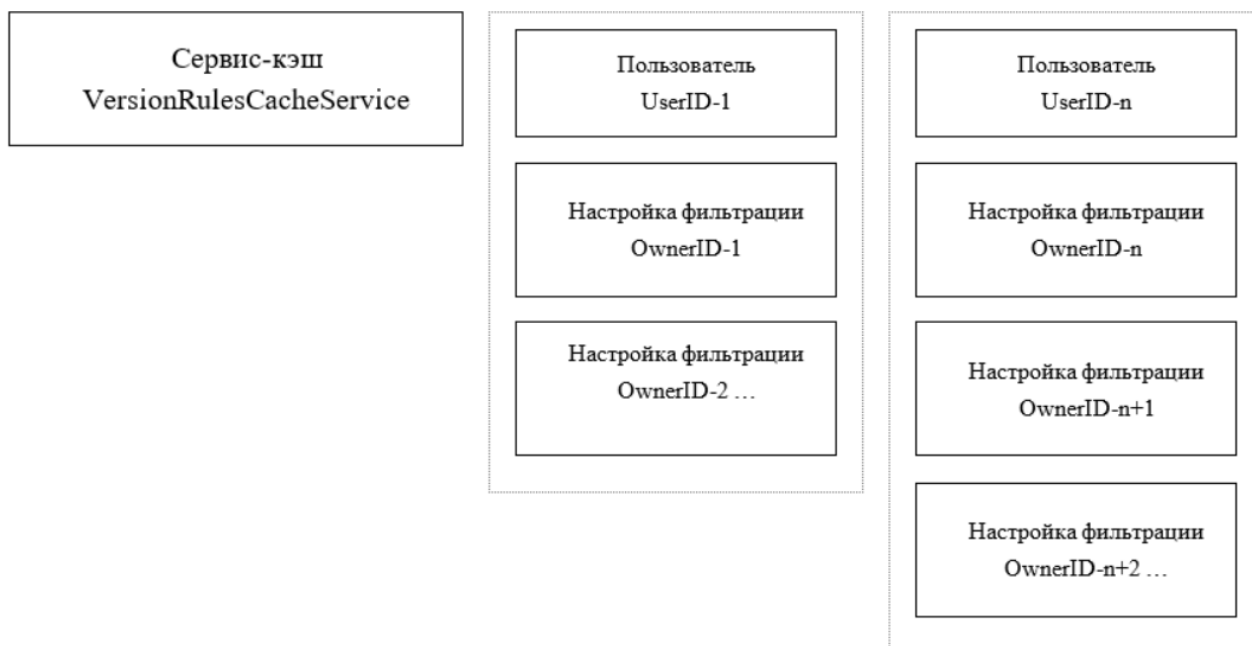
Для того, чтобы хранить множество настроек фильтрации (например, для разных окон «Универсального клиента»), а также для управления этими настройками, используются сервис-кэш `VersionRulesCacheService` и классы, реализующие интерфейс **`IFiltrationSettings`**.

Каждый вариант настроек фильтрации имеет в своём составе:

- уникальный идентификатор `OwnerID` (Guid) в виде строки – ключ настроек фильтрации;
- тип правила подбора версий (одно из значений перечислителя `VersionsRuleType`);
- `[F_ОБЪЕКТ_ID]` правила подбора версий (для стандартного правила подбора версий);
- индекс варианта значений правила (если в составе критериев правила есть переменные).

Сервис `Intermech.Kernel.VersionRulesCacheService` управляет коллекцией настроек фильтрации по двум параметрам - `UserID` пользователя и `OwnerID` настроек. Таким образом, во всех методах, в которых требуются настройки фильтрации, необходимо использовать в качестве дополнительного параметра уникальный идентификатор настроек фильтрации – `OwnerID`. Значение `UserID` получается из одноименного свойства интерфейса `IUserSession`.

В результате получается следующая коллекция настроек фильтрации для всех пользователей системы:



В результате появляется возможность с помощью одного текстового ключа управлять настройками фильтрации. При этом имеется возможность в будущем нарастить функционал настроек фильтрации.

6.3.3.1 Управление настройками фильтрации состава

Для управления настройками фильтрации пользователем есть два способа - с помощью тулбара **Фильтрация состава**, который расположен на главной форме «Универсального клиента», а также с помощью формы `Intermech.PropertyEditors.FiltrationSettingsForm`, которая позволяет выбирать правила подбора версий для указанной настройки фильтрации, а также управлять вариантами значений переменных для этих правил.

6.3.4 Делегат `FiltrationChanged`

Делегат для событий от службы `IFiltrationService`.

6.3.4.1 Пространство имён

`Intermech.Interfaces.Client`

6.3.4.2 Сигнатура

Синтаксис:

```
public delegate void FiltrationChanged (IFiltrationSettings NewFiltration, bool FiltrationValid);
```

6.3.5 Перечислитель `VersionRuleType`

Тип правила подбора версий.

6.3.5.1 Пространство имён

`Intermech.Interfaces`

6.3.5.2 Значения перечислителя

`vrtStandardRule = 0`

Правило подбора версий стандартное.

`vrtLatestVersionsRule = -1`

Правило, возвращающее последние версии объектов. Создаётся сервисом-кэшем правил.

`vrtAllVersionsRule = -2`

Правило, возвращающее все версии объектов. Создаётся сервисом-кэшем правил.

`vrtSystemRule = -3`

Правило системное. Создаётся сервисом-кэшем правил.

6.3.6 Интерфейс `IFiltrationClass`

Интерфейс должен быть реализован в окнах универсального клиента, которым требуются персональные настройки фильтрации составов и поддержка работы с панелью инструментов «Фильтрация составов».

6.3.6.1 Пространство имён

Intermech.Interfaces.Client

6.3.6.2 Свойства интерфейса

FiltrationOwnerID

Уникальный строковый идентификатор настроек фильтрации составов.

Синтаксис:

```
string FiltrationOwnerID { get; }
```

Если окно реализует данный интерфейс, то при его активации для этого окна создаются его персональные настройки фильтрации составов. Доступ к этим настройкам осуществляется по ключу **FiltrationOwnerID**. Настройки хранятся на сервере приложений в службе **IVersionRulesCacheService**. Когда окно закрывается и при этом не требуется сохранять его настройки, окно должно позаботиться о том, чтобы удалить свои настройки (обратившись к соответствующему методу данной серверной службы). Если окно не удаляет настройки, а также в течение 60 дней к настройкам не происходит ни одного обращения, серверная служба **IVersionRulesCacheService** автоматически удалит указанные настройки из базы данных (сборка мусора во избежание его накопления в настройках пользователей).

6.3.7 Интерфейс IFiltrationSettings

Интерфейс предназначен для хранения настроек фильтрации составов. Реализация осуществляется объектом FiltrationSettings.

6.3.7.1 Пространство имён

Intermech.Interfaces

6.3.7.2 Способы получения интерфейса

Интерфейс можно получить у службы **Intermech.Interfaces.Client.IFiltrationService**.

6.3.7.3 Свойства интерфейса

CurrentRule

Экземпляр текущего правила подбора версий. Поле заполняется только тогда, когда правило подбора не связано с объектом **Правило** из БД, а назначается из группирующего объекта, т.п.

Синтаксис:

```
VersionRule CurrentRule { get; set; }
```

CurrentRuleVars

Индекс основного варианта значений переменных для текущего правила подбора версий. Значение актуально только если правило подбора версий является параметризованным (содержит переменные).

Синтаксис:

```
Int32 CurrentRuleVars { get; set; }
```

LastAccess

Дата и время последнего доступа к настройкам фильтрации.

Синтаксис:

```
DateTime LastAccess { get; }
```

OwnerID

Уникальный объект-ключ владельца данных настроек фильтрации составов.

Синтаксис:

```
string OwnerID { get; }
```

RuleVars

Ссылка на коллекцию значений [Ключ]=[Значение], где [Ключ] - это значение OBJECT_ID правила подбора версий, а [Значение] - это номер основного варианта значений переменных для этого правила.

Синтаксис:

```
Dictionary<Int64, Int32> RuleVars { get; }
```

Tags

Ссылка на коллекцию сериализуемых значений [Ключ]=[Значение] где ключ - Guid в виде строки, а значение – любой сериализуемый объект. В тегах хранятся какие-либо данные, которые надо "прицепить" к текущим настройкам фильтрации. С помощью данного словарика в параметры запроса передаются

дополнительные ключи, управляющие, например, фильтрацией составов по контекстам, по допустимым заменам, т.п.

Синтаксис:

```
HybridDictionary Tags { get; }
```

6.3.8 Интерфейс IFiltrationService

Интерфейс управляет настройками фильтрации в окнах **Навигатора**, управляет панелью инструментов **Фильтрация состава** в главной форме, а также предоставляет возможность добавлять на эту панель элементы управления.

6.3.8.1 Пространство имён

Intermech.Interfaces.Client

6.3.8.2 Способы получения интерфейса

ServicesManager.GetService(typeof(IFiltrationService)) as IFiltrationService

6.3.8.3 Свойства интерфейса

Filtration

Текущие настройки фильтрации составов.

Синтаксис:

```
IFiltrationSettings Filtration { get; set; }
```

FiltrationRuleID

Идентификатор объекта текущего правила подбора версий. Если правило не связано с объектом базы данных, то данное свойство возвращает **0**.

Синтаксис:

```
Int64 FiltrationRuleID { get; }
```

FiltrationServiceOwnerID

Уникальный идентификатор владельца текущих настроек фильтрации. Изменение владельца ведёт к перечисыванию настроек фильтрации состава. Автоматически будет вызвано событие, уведомляющее о смене в настройках фильтрации состава.

Синтаксис:

```
FiltrationServiceOwnerID { get; set; }
```

FiltrationToolbarHidden

Свойство, определяющее скрытость тулбара **Фильтрация состава**.

Синтаксис:

```
bool FiltrationToolbarHidden { get; set; }
```

FiltrationToolbarVisible

Свойство, определяющее видимость тулбара **Фильтрация состава**.

Синтаксис:

```
bool FiltrationToolbarVisible { get; set; }
```

RuleClass

Текущее правило фильтрации состава, информация по которому отображена в тулбаре **Фильтрация состава**. Причём если у правила есть переменные, то они уже заданы согласно текущей настройке фильтрации. Можно задавать свои экземпляры правил подбора версий, причём не связанные с реальными объектами из базы данных, например, правила подбора из группирующих объектов.

Синтаксис:

```
VersionsRule RuleClass { get; set; }
```

RuleCompatible

Если выбранное правило является вариантом значений переменных (т.е. создано на базе родительского правила), то это поле отражает, совместимо ли правило с родительским вариантом (на случай, если были изменения в родительском правиле после создания вариантов его значений переменных).

Синтаксис:

```
bool RuleCompatible { get; }
```

RuleErrorCode

Код ошибки для текущего правила:

- 0 – правило не выбрано,
- 1 – настройки недействительны - правило было изменено,
- 2 – нет ошибок, правило настроено,
- 3 – нет вариантов значений переменных для правила,
- 4 – фильтрация состава выключена (зарезервировано, не используется),
- 5 – не указан основной вариант значений переменных,
- 6 – правило является некорректным.

Синтаксис:

```
bool RuleErrorCode { get; }
```

RuleValid

Корректно ли выбранное правило подбора версий (для проверки выполняется метод Valid правила, а также проверяется наличие у него переменных значений). Если правило не корректно, его нельзя использовать для фильтрации составов.

Синтаксис:

```
bool RuleValid { get; }
```

6.3.8.4 События интерфейса

OnFiltrationChanged

Событие вызывается после смены настроек фильтрации состава на новое значение.

Синтаксис:

```
public event FiltrationChanged OnFiltrationChanged
```

6.3.8.5 Методы интерфейса

AddNewButton

Метод создаёт и добавляет новую кнопку на панель инструментов «Фильтрация составов».

Синтаксис:

```
Intermech.Bars.ButtonItem AddNewButton ();
```

Результат:

Созданная кнопка.

AddNewCombobox

Метод создаёт и добавляет новый выпадающий список (комбо-бокс) на панель инструментов **Фильтрация составов**.

Синтаксис:

```
Intermech.Bars.ComboBoxItem AddNewCombobox ();
```

Результат:

Созданный выпадающий список.

FiltrationApplyUpdates

Метод применяет все настройки фильтрации из панели инструментов **Фильтрация составов** к текущему окну **Навигатора**. При этом генерируется событие **OnFiltrationChanged**.

Синтаксис:

```
void FiltrationApplyUpdates ();
```

FiltrationUpdate

Метод перечитывает текущие настройки фильтрации состава, обновляет тулбар **Фильтрация состава**. При необходимости генерирует событие **OnFiltrationChanged**.

Синтаксис:

```
void FiltrationUpdate (bool FireEvent);
```

Параметры:

FireEvent

Если равно true, то генерируется событие **OnFiltrationChanged**.

FiltrationUpdateCompatibility

Метод обновляет на панели инструментов **Фильтрация составов** состояние элементов управления, которые не поддерживают интерфейс **ICommandState**.

Синтаксис:

```
void FiltrationUpdateCompatibility ();
```

6.3.9 Класс CompareFunctionsHelper

Вспомогательный класс для работы со списком функций сравнения.

6.3.9.1 Пространство имён

Intermech.Interfaces

Список имён функций для сравнения приведён в таблице в разделе **Функции сравнения**.

6.3.10 Класс ComparableValue

Класс хранит в себе значение для сравнения, а также тип этого значения. Реализует интерфейс ICloneable. Используется в критериях подбора версий.

6.3.10.1 Пространство имён

Intermech.Interfaces

6.3.10.2 Свойства класса

Attribute

Атрибут, значения которого используется в критериях подбора версий.

Синтаксис:

```
public MyAttributeMetadata Attribute { get; }
```

AttrType

Тип данных, который хранится в атрибуте.

Синтаксис:

```
public FieldTypes AttrType { get; set; }
```

Value

Значение атрибута. Если свойство ValueType равно **CompareTypesHelper.ctAttribute**, то данное значение равно Guid атрибута.

Синтаксис:

```
public object Value { get; }
```

ValueType

Вид значения для сравнения (константа класса **CompareTypesHelper**).

Синтаксис:

```
public string ValueType { get; }
```

6.3.11 Класс VersionRuleCriterion

Класс представляет собой критерий правила подбора версий. Реализует интерфейс ICloneable.

6.3.11.1 Пространство имён

Intermech.Interfaces

6.3.11.2 Свойства класса

BoolFunction

Логическая функция для объединения данного критерия со следующим критерием в составе правила подбора версий.

Синтаксис:

```
public string BoolFunction { get; set; }
```

CFunc

Вспомогательный класс по работе со списком функций сравнения и пр.

Синтаксис:

```
public CompareFunctionsHelper CFunc { get; }
```

ComparableValues

Список значений для сравнения, которыми оперирует критерий подбора.

Синтаксис:

```
public List<ComparableValue> ComparableValues { get; }
```


6.3.12 Класс VersionsRule

Класс представляет собой правило подбора версий. Класс реализует интерфейсы ICloneable, IEnumerator, IEnumerable.

6.3.12.1 Пространство имён

Intermech.Interfaces.VersionsRule

6.3.12.2 Свойства класса

ActualDate

Дата для получения состава. Если равна **DateTime.MinValue**, то не свойство используется во время подбора версий.

Синтаксис:

```
public DateTime ActualDate { get; set; }
```

Criteriaes

Список критериев подбора.

Синтаксис:

```
public List<VersionRuleCriterion> Criteriaes { get; }
```

CurrentRuleType

Тип правила подбора версий.

Синтаксис:

```
public VersionsRuleType CurrentRuleType { get; set; }
```

6.4 Примеры

6.4.1 Создание обработчиков коллекций связей определенных типов

1. Пишем класс-обработчик собственного списка связей, порождая его от Intermech.Kernel.DBRelationCollection и реализуя в нем (при необходимости) свои дополнительные интерфейсы.

```
class MyDBRelationCollection: DBRelationCollection, IMyDBRelationCollection
{
    override ObjectMethod() // Перекрываем имеющиеся методы
    {
    }
    public MyMethod() // И реализуем методы своих интерфейсов
    {
    }
}
```

2. Выбираем Guid MyGuid для своего типа связей и создаем этот тип в базе данных.
3. Пишем создатель списков связей данного типа и реализуем в нем поддержку интерфейса Intermech.Interfaces.IDBRelationCollectionCreator.

```
class MyRelationCollectionCreator: IDBRelationCollectionCreator
{
    public IDBRelationCollection CreateRelationCollection(IUserSession uSession, Guid guid, int relationTypeID)
    {
        if (guid == MyGuid) // Это если Ваш создатель поддерживает много типов связей (а не только //MyGuid). Но можно писать свой отдельный создатель для каждого типа объектов.
        {
            return new MyDBRelationCollection(uSession, int relationType);
        }
    }
}
```

4. При загрузке своего плагина регистрируем свой создатель коллекций связей для типа MyGuid.

...

```
IDBRelationCollectionCreator creator = new MyRelationCollectionCreator();
ICreatorContainer creators = ServerServices.GetService(typeof(IDBRelationCollectionService)) as
ICreatorContainer;
creators.AddCreator(MyGuid, creator);
```

7.1 Обработка событий

7.1.1 Интерфейс IEventLogHelper

Интерфейс службы, обрабатывающей различные события, генерируемые ядром системы. Служба также содержит методы записи событий в журнал или log-файлы.

7.1.1.1 Пространство имен

Intermech.Interfaces.Server

7.1.1.2 События интерфейса:

Intermech.Interfaces.Server.AfterAddEventHandler AfterAddEvent

Событие добавления информации в журнал, возникающее после записи в журнал.

Intermech.Interfaces.Server.CacheReloadHandler AfterCacheReload

Событие генерируется сразу после перезагрузки кэша метаданных из БД. Рекомендуется подписываться на данное событие для перезагрузки собственных кэшей на сервере приложений.

Intermech.Interfaces.Server.ObjectTypeChangeHandler AfterChangeObjectTypeEvent

Событие наступает после смены типа объекта Sender. Вторым параметром выступает идентификатор предыдущего типа данного объекта.

Intermech.Interfaces.Server.ObjectEventHandler AfterCheckinEvent

Событие возникает после возврата изменений объекта в базу.

Intermech.Interfaces.Server.ObjectEventHandler AfterCheckoutEvent

Событие возникает после взятия объекта на изменение.

Intermech.Interfaces.Server.AfterCreateAttributeTypeHandler AfterCreateAttributeTypeEvent

Событие генерируется после создания в системе нового атрибута-типа.

Intermech.Interfaces.Server.CreateRelationHandler AfterCreateRelationExEvent

Событие генерируется после создания связи.

Intermech.Interfaces.Server.DeleteAttributeTypeHandler AfterDeleteAttributeTypeEvent

Событие генерируется после удаления атрибута-типа (вызывается после коммита).

Intermech.Interfaces.Server.DeleteObjectTypeHandler AfterDeleteObjectTypeEvent

Событие генерируется после удаления типа объектов (вызывается после коммита).

Intermech.Interfaces.Server.DeleteRelationHandler AfterDeleteRelationEvent

Событие генерируется в транзакции сразу после удаления связи sender (deleteMode зарезервировано).

Intermech.Interfaces.Server.LoginHandler AfterLoginEvent

Событие происходит в момент подключения к серверу нового пользователя.

Intermech.Interfaces.Server.LoginHandler AfterLogoutEvent

Событие происходит в момент отключения пользователя от сервера.

Intermech.Interfaces.Server.NextLCStepHandler AfterNextLCStepEvent

Событие наступает после перевода объекта на следующий этап ЖЦ.

Intermech.Interfaces.Server.ObjectEventHandler AfterPurgeObjectEvent

Событие возникает после физического удаления версии объекта из базы данных.

Intermech.Interfaces.Server.ObjectEventHandler AfterSaveChangesEvent

Событие возникает после сохранением изменений объекта в базу.

Intermech.Interfaces.Server.ObjectEventHandler AfterUndoCheckoutEvent

Событие возникает после отмены изменений в объекте.

Intermech.Interfaces.Server.BeforeAddEventHandler BeforeAddEvent

Событие генерируется перед добавлением записи в журнал событий. Позволяет, например, сгенерировать исключение для отмены действия, вызвавшего данное событие. На вызов данного события не влияют настройки записи журнала событий – даже если запись в журнал указанной категории событий администратором запрещена, в данный обработчик управление все равно попадает.

Делегат, описывающий событие, которое срабатывает перед добавлением записи в журнал регистрации событий.

```
public delegate void BeforeAddEventHandler(Int64 ObjectID, Int64 RelationID, int CategoryType, Int64 CategoryID, string ObjectName, string Note, ActionType EventType, EventlogRecordType AuditType, Int64 UserID, string ComputerName, IUserSession aSession);
```

Параметры:

ObjectID

Идентификатор версии объекта, к которой относится данное событие. Содержит 0 для событий, не относящихся к информационным объектам.

RelationID

Идентификатор связи, к которой относится данное событие. Если к связи событие отношения не имеет, то значение параметра равно 0.

CategoryType

Идентификатор категории, к которой относится событие.

CategoryID

Идентификатор экземпляра данной категории.

ObjectName

Наименование категории и/или объекта, над которым совершено данное событие (например, Атрибут **Материал** объекта **Деталь 124 - 3214432-01 Вал**)

Note

Комментарии к событию. Может использоваться, например, для хранения предыдущих значений параметров.

EventType

Тип события.

AuditType

Тип записи в журнале событий (ошибка, предупреждение или информация).

UserID

Идентификатор пользователя, инициировавшего событие.

ComputerName

Имя компьютера, с которого инициировали событие.

aSession

Сессия, инициировавшая событие. Если работа идет вне контекста сессии, значение параметра может быть равным **null**.

[Intermech.Interfaces.Server.ObjectTypeChangeHandler BeforeChangeObjectTypeEvent](#)

Событие наступает перед сменой типа объекта Sender. Вторым параметром выступает идентификатор нового типа объекта.

[Intermech.Interfaces.Server.ObjectEventHandler BeforeCheckinEvent](#)

Событие возникает перед возвратом изменений объекта в базу.

[Intermech.Interfaces.Server.ObjectEventHandler BeforeCheckoutEvent](#)

Событие возникает перед взятием объекта на изменение.

[Intermech.Interfaces.Server.BeforeCreateRelationHandler BeforeCreateRelationEvent](#)

Событие генерируется перед созданием связи между объектами.

[Intermech.Interfaces.Server.DeleteAttributeTypeHandler BeforeDeleteAttributeTypeEvent](#)

Событие генерируется перед удалением атрибута-типа (в транзакции).

[Intermech.Interfaces.Server.DeleteObjectTypeHandler BeforeDeleteObjectTypeEvent](#)

Событие генерируется перед удалением типа объектов (в транзакции).

[Intermech.Interfaces.Server.DeleteRelationHandler BeforeDeleteRelationEvent](#)

Событие генерируется перед удалением связи sender (deleteMode зарезервировано).

[Intermech.Interfaces.Server.NextLCStepHandler BeforeNextLCStepEvent](#)

Событие наступает перед переводом объекта на следующий этап ЖЦ.

[Intermech.Interfaces.Server.ObjectEventHandler BeforeObjectPrintEvent](#)

Событие наступает перед выводом на печать.

[Intermech.Interfaces.Server.ObjectEventHandler BeforeObjectSaveToDiskEvent](#)

Событие наступает перед сохранением на диск.

[Intermech.Interfaces.Server.ObjectEventHandler BeforePurgeObjectEvent](#)

Событие наступает перед физическим удалением версии объекта из базы данных.

Intermech.Interfaces.Server.BeforeRecordsSelectHandler BeforeRecordsSelectEvent

Событие генерируется перед получением записей в методе IDBRecords.Select и позволяет модифицировать условия получения записей.

Intermech.Interfaces.Server.ObjectEventHandler BeforeSaveChangesEvent

Событие возникает перед сохранением изменений объекта в базу.

Intermech.Interfaces.Server.ObjectEventHandler BeforeUndoCheckoutEvent

Событие возникает перед отменой изменений в объекте.

Intermech.Interfaces.Server.ObjectEventHandler CommitCreationObjectEvent

Событие возникает в функции DBObject.CommitCreation, завершающей создание объекта.

Intermech.Interfaces.Server.CreateAttributeHandler CreateAttributeEvent

Событие генерируется в транзакции после создания экземпляра атрибута.

Intermech.Interfaces.Server.ObjectEventHandler CreateObjectEvent

Событие возникает сразу после создания заготовки объекта sender. Если сгенерировать исключение, то заготовка будет удалена транзакцией.

Intermech.Interfaces.Server.GetAttributeDefaultValueHandler GetAttributeDefaultValueEvent

Событие генерируется при получении значения по умолчанию для создаваемых экземпляров атрибутов.

Intermech.Interfaces.Server.GetObjectSecurityHandler GetObjectSecurity

Событие получения списка дополнительных интерфейсов безопасности объекта.

Intermech.Interfaces.Server.GetRecordsListHandler GetRecordsListEvent

Событие генерируется в функции Select, которая получает список объектов, связей и т.п. в соответствии с условиями parameters. В событии можно провести модификацию таблицы возвращаемых данных.

Intermech.Interfaces.Server.OnCloseEventHandler OnCloseEvent

Завершение события в журнале.

Делегат, описывающий событие, которое срабатывает после закрытия в журнале события номер EventID.

```
public delegate void OnCloseEventHandler(Int64 EventID, string Note, EventlogRecordType AuditType, IUserSession aSession);
```

Параметры:

EventID

Идентификатор завершаемого события.

Note

Комментарии к событию.

AuditType

Тип записи в журнале.

aSession

Сессия, инициировавшая событие. Может быть null, если работа идет вне контекста сессии.

Intermech.Interfaces.Server.OnCloseEventHandler2 OnCloseEvent2

Завершение события в журнале.

7.1.1.3 Методы интерфейса:

AddActionName

Добавить наименование действия actType для категории categoryType. Для объектов в качестве categoryID следует передавать ObjectType.

Синтаксис:

```
void AddActionName (Int32 categoryType; Int64 categoryID; Intermech.ActionType actType; string actionName)
```

Параметры:

categoryType

Идентификатор категории.

categoryID

Идентификатор экземпляра данной категории.

actType

Тип действия.

actionName

Наименование действия.

AddAttributeWriteHandler

Добавить обработчик события записи значения атрибута attributeID. В качестве attributeID можно послать имя или Guid атрибута. Если attributeID==0, то событие будет сгенерировано для любого атрибута. В attributeHandler можно записать свое значение вместо записываемого в базу.

Синтаксис:

```
void AddAttributeWriteHandler (object attributeID;  
Intermech.Interfaces.Server.WriteAttributeValueHandler attributeHandler)
```

Параметры:

attributeID

Идентификатор атрибута (String, Int32 или Guid).

attributeHandler

Делегат, позволяющий записать произвольное значение атрибута в базу вместо текущего.

AddEvent

Добавить новое событие в журнал. Возвращает идентификатор добавленного события.

Синтаксис:

```
Int64 AddEvent (Int64 ObjectID; Int64 RelationID; Int32 CategoryType; Int64 CategoryID;  
string ObjectName; string Note; Intermech.ActionType EventType; Intermech.EventlogRecordType AuditType;  
Int64 UserID; string ComputerName; Intermech.Interfaces.IUserSession aSession)
```

Параметры:

ObjectID

Идентификатор версии объекта, к которой относится данное событие. Например, если мы изменяем значение атрибута, то в F_CATEGORY_ID будет записан идентификатор атрибута, а в F_OBJECT_ID - идентификатор объекта, у которого изменяют атрибут. Если событие относится собственно к объекту (например, удаляем сам объект), то мы F_CATEGORY_ID = F_OBJECT_ID. Если событие не касается объекта, то ObjectID=0.

RelationID

Идентификатор связи, к которой относится данное событие. Если к связи событие отношения не имеет, то RelationID==0.

CategoryType

Идентификатор категории (например, атрибут=3).

CategoryID

Идентификатор экземпляра данной категории (например, AttributeID).

ObjectName

Наименование категории и/или объекта, над которым совершено данное событие (например, Атрибут **Материал** объекта **Деталь 124-3214432-01 Вал**).

Note

Комментарии. Может использоваться, например, для хранения предыдущих значений параметров.

EventType

Тип события.

AuditType

Тип записи в журнале событий.

UserID

Идентификатор пользователя, инициировавшего событие.

ComputerName

Имя компьютера, с которого инициировали событие.

aSession

Сессия, инициировавшая событие. Может быть null, если работы идет вне контекста сессии.

AddToTrace

Записать строку EventStr в файл-протокол.

Синтаксис:

```
Int32 AddToTrace (string EventStr; Int32 TraceLevel; string TraceFileName)
```

Параметры:

EventStr

Строка для записи в протокол.

TraceLevel

Уровень трассировки, при котором будет записана данная строка (например, если TraceLevel = Consts.traceAlways, то запись будет идти всегда, а уровень Consts.traceError используется для записи ошибок).

TraceFileName

Имя файла трассировки. Если не указывать, то будет imserver.log. Если путь не указан, то файл складывается в один каталог с главным файлом трассировки imserver.log.

CloseEvent

Завершить указанное событие.

Синтаксис:

```
Int32 CloseEvent (Int64 EventID; Intermech.EventlogRecordType AuditType; string Note;  
Intermech.Interfaces.IUserSession aSession)
```

Параметры:

EventID

Идентификатор завершаемого события.

AuditType

Тип записи в журнале можно изменить при закрытии (например, был AccesDenied, а стал - AccessGranted).

Note

Можно изменить комментарии.

aSession

Сессия, инициировавшая событие. Может быть null, если работы идет вне контекста сессии.

CloseEvent

Завершить указанное событие.

Синтаксис:

```
Int32 CloseEvent (Int64 EventID; Int64 ObjectID; Int64 CategoryID; string ObjectName;  
string Note; Intermech.EventlogRecordType AuditType;  
Intermech.Interfaces.IUserSession aSession)
```

Параметры:

EventID

Идентификатор завершаемого события.

ObjectID

Идентификатор версии объекта, к которой относится данное событие.

CategoryID

Идентификатор экземпляра данной категории.

ObjectName

Наименование категории и/или объекта, над которым совершено данное событие.

Note

Можно изменить комментарии.

AuditType

Тип записи в журнале можно изменить при закрытии например, был AccesDenied, а стал - AccessGranted).

aSession

Сессия, инициировавшая событие. Может быть null, если работы идет вне контекста сессии.

GetActionName

Возвращает наименование действия actType для категории categoryType и объекта categoryID. Если для categoryID>0 наименование не найдено, то идет поиск для categoryID=0. Если для categoryType>0 наименование не найдено, то идет поиск для categoryType=0 и categoryID=0.

Синтаксис:

```
string GetActionName (Int32 categoryType; Int64 categoryID; Intermech.ActionType actType)
```

Параметры:

categoryType

Идентификатор категории.

categoryID

Идентификатор экземпляра данной категории.

actType

Тип действия.

Возвращаемое значение:

Наименование действия.

GetAttributeID

Возвращает числовой идентификатор атрибута по его имени, Guid или числовому идентификатору [attributeID].

Синтаксис:

```
Int32 GetAttributeID (object attributeID)
```

Параметры:

attributeID

Имя, Guid или числовой идентификатор атрибута.

GetAttributeID

Возвращает числовой идентификатор атрибута по его имени, Guid или числовому идентификатору [attributeID].

Синтаксис:

```
Int32 GetAttributeID (object attributeID; bool failIfNotFound)
```

Параметры:

attributeID

Имя, Guid или числовой идентификатор атрибута.

failIfNotFound

true – генерировать исключение, если атрибут с указанным идентификатором не будет найден.

RemoveAttributeWriteHandler

Удаляет обработчик события записи значения атрибута attributeID, зарегистрированный функцией AddAttributeWriteHandler.

Синтаксис:

```
void RemoveAttributeWriteHandler (object attributeID;  
Intermech.Interfaces.Server.WriteAttributeValueHandler attributeHandler)
```

Параметры:

attributeID

Идентификатор атрибута (String, Int32 или Guid).

attributeHandler

Делегат, позволяющий записать произвольное значение атрибута в базу вместо текущего.

TruncateTraceFile

Обрезает trace-файл tracefileName если его размер превысил Consts.traceFileMaxSize. Возвращает размер log-файла.

Синтаксис:

```
Int32 TruncateTraceFile (string tracefileName)
```

Параметры:

tracefileName

trace-файл.

Возвращаемое значение:

Текущий размер файла.

TruncateTraceFile

Удаляет trace-файл tracefileName, если его размер превысил tracefileSize. Возвращает размер log-файла.

Синтаксис:

```
Int32 TruncateTraceFile (string tracefileName; Int32 tracefileSize)
```

Параметры:

tracefileName

trace-файл.

tracefileSize

Предельный размер trace-файла.

Возвращаемое значение:

Текущий размер файла.

7.1.2 Интерфейс IEventLog

Интерфейс для работы с журналом событий.

7.1.2.1 Пространство имен

Intermech.Interfaces

7.1.2.2 Реализуемые интерфейсы:

Intermech.Interfaces.IDBRecords

Intermech.Interfaces.IDBSessionable

7.1.2.3 Свойства интерфейса:

Intermech.Interfaces.EventlogSettings Settings { get; set; }

Настройки регистрации событий.

7.1.2.4 Методы интерфейса:

AddToTrace

Добавить запись в лог-файл.

Синтаксис:

```
Int32 AddToTrace (string EventStr; Int32 TraceLevel; string TraceFileName)
```

Параметры:

EventStr

Строка, добавляемая в лог-файл.

TraceLevel

Номер уровня детализации трассировки, начиная с которого запись попадет в лог-файл.

TraceFileName

Имя файла трассировки, в который нужно добавить эту строку. Если пусто, то имя imserver.log. Если путь не указан, то файл складывается в один каталог с главным файлом трассировки imserver.log.

ClearEvents

Очищает журнал событий старше даты fromDate.

Синтаксис:

```
void ClearEvents (System.DateTime fromDate)
```

Параметры:

fromDate

Дата, до которой события считаются устаревшими.

DeleteEvents

Удалить события.

Синтаксис:

```
void DeleteEvents (Int64[] EventsID)
```

Параметры:

EventsID

Массив с идентификаторами событий, которые нужно удалить.

GetActionNamesHash

Возвращает хэш с расшифровкой имен событий, генерируемых в системе.

Синтаксис:

```
System.Collections.Hashtable GetActionNamesHash ()
```

Возвращаемое значение:

Хэш с расшифровкой имен событий, генерируемых в системе.

Select

Получить таблицу со списком событий, соответствующих paramSet.

Синтаксис:

```
System.Data.DataTable Select (Intermech.Kernel.Search.DBRecordSetParams paramSet; bool translateValues)
```

Параметры:

paramSet

Условия запроса в журнал событий.

translateValues

Если translateValues - true, то поля событий будут расшифрованы (имена пользователей, названия действий и т.п.).

Возвращаемое значение:

Таблица со списком подходящих событий.

Ниже приведена структура таблицы событий с описанием допустимых полей.

Таблица IMS_EVENTLOG. Журнал событий.

Имя поля	Тип поля	Комментарий
F_EVENT_ID	BIGINT	Уникальный идентификатор события.
F_CATEGORY_TYPE	INTEGER	Идентификатор категории данных, к которой относится событие (объекты, связи, атрибуты, типы объектов и пр.)
F_CATEGORY_ID	BIGINT	Идентификатор экземпляра указанной категории.
F_OBJECT_ID	BIGINT	Идентификатор версии объекта, к которой относится событие (содержит 0 если событие не относится к объектам).
F_OBJECT_NAME	NVARCHAR(250)	Имя объекта или категории данных, к которой относится событие.
F_USER_ID	BIGINT	Идентификатор пользователя, инициировавшего событие.
F_COMPUTER_NAME	NVARCHAR(20)	Имя компьютера, с которого было инициировано событие.
F_NOTE	NVARCHAR(450)	Комментарии к событию.
F_EVENT_TYPE	INTEGER	Тип действия, к которому относится данное событие.
F_BEGIN_DATE	DATETIME	Дата и время начала события.
F_END_DATE	DATETIME	Дата и время завершения события.
F_AUDIT_TYPE	INTEGER	Тип записи в журнале (0 – права предоставлены, 1 – в доступе отказано, 2 – уведомление, 3 – предупреждение, 4 – ошибка).
F_RELATION_ID	BIGINT	Идентификатор связи, к которой относится событие (0 если событие не относится к связям между объектами).

7.1.3 Структура EventlogSettings

Структура для передачи и хранения настроек регистрации событий.

7.1.3.1 Пространство имен

Intermech.Interfaces

7.1.3.2 Поля структуры:

bool AutoClear

Автоматическая очистка журнала.

bool LogOn

Включена/выключена регистрация всех событий.

Int64[] NotLoggedObjects

Идентификаторы объектов, для которых не требуется проводить регистрацию событий.

Int32[] NotLoggedTypes

Идентификаторы типов, для которых не проводить регистрацию событий.

Int32 RecordsKeepDays

Сколько дней сохранять записи журнала.

7.1.3.3 Методы структуры:

EventlogSettings (конструктор)

Создать экземпляр структуры для передачи и хранения настроек регистрации событий.

Синтаксис:

```
void EventlogSettings (bool logOn; Int64[] notLoggedObjects; Int32[] notLoggedTypes;  
bool autoClear; Int32 recordsKeepDays)
```

Параметры:

logOn

Включена/выключена регистрация всех событий.

notLoggedObjects

Идентификаторы объектов, для которых не требуется проводить регистрацию событий.

notLoggedTypes

Идентификаторы типов, для которых не проводить регистрацию событий.

autoClear

Автоматическая очистка журнала.

recordsKeepDays

Сколько дней сохранять записи журнала.

7.1.4 Примеры

7.1.4.1 Как подписаться на события, генерируемые ядром серверной части системы

Все события реализованы в интерфейсе Intermech.Interfaces.Server.IEventLogHelper.

1. Описываем метод обработки события (например, удаления атрибута-типа из базы данных).

```
public void DeleteAttributeType(IDBAttributeType sender)
{
    // Что-то делаем
}
```

2. Подключаем свой обработчик события.

```
IEventLogHelper ehelper = ServerServices.GetService(typeof(IEventLogHelper)) as IEventLogHelper;
ehelper.AfterDeleteAttributeTypeEvent += new DeleteAttributeTypeHandler(DeleteAttributeType);
```

3. При выгрузке своего модуля отключаем обработчик.

```
IEventLogHelper ehelper = ServerServices.GetService(typeof(IEventLogHelper)) as IEventLogHelper;
ehelper.AfterDeleteAttributeTypeEvent -= new DeleteAttributeTypeHandler(DeleteAttributeType);
```

7.2 Работа с выборками и классификаторами

7.2.1 Интерфейс ISelectionsService

Интерфейс для работы с выборками и классификаторами.

7.2.1.1 Пространство имен

Intermech.Interfaces

7.2.1.2 Методы интерфейса:

AddClassifierToCache

Добавить классификатор в кэш.

Синтаксис:

```
void AddClassifierToCache (Intermech.Interfaces.IUserSession session; Int64 classifierID)
```

Параметры:

session

Сессия.

classifierID

Идентификатор версии объекта-классификатора.

CanUpperMemo

Возвращает true, если СУБД поддерживает приведение мемо-полей к верхнему регистру.

Синтаксис:

```
bool CanUpperMemo (object userSession)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

Возвращаемое значение:

true, если СУБД поддерживает приведение мемо-полей к верхнему регистру.

ClearCashe

Очистка кэша условий выборок.

Синтаксис:

```
void ClearCashe ()
```

DeleteClassifierFromCache

Удалить классификатор из кэша.

Синтаксис:

```
void DeleteClassifierFromCache (Int64 classifierID)
```

Параметры:

classifierID

Идентификатор версии удаляемого объекта-классификатора.

DisableConditionStructures

Установить в кэше массив с индексами отключенных условий для выборки.

Синтаксис:

```
void DisableConditionStructures (Int64 selectionID; System.Collections.Generic.List<Int32> conditionIndexes)
```

Параметры:

selectionID

ID выборки.

conditionIndexes

Массив с индексами отключенных условий.

ExcludeObjects

Исключение объектов из классификатора (ручной выборки).

Синтаксис:

```
void ExcludeObjects (object userSession; Int64 selectionID; Int64[] objectIDs)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

selectionID

Идентификатор классификатора (ручной выборки).

objectIDs

Массив идентификаторов объектов, которые надо исключить.

ExistsObject

Проверка наличия объекта в классификаторе (ручной выборке).

Синтаксис:

```
bool ExistsObject (object userSession; Int64 selectionID; Int64 objectID)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

selectionID

Идентификатор классификатора (ручной выборки).

objectID

Идентификатор объекта, наличие которого проверяется.

Возвращаемое значение:

true – указанный объект найден в классификаторе (выборке).

GenerateNextClassifierKey

Получить следующее доступное значение для ключа классификатора (каталога Imbase либо другого объекта с атрибутом **ключ папки классификатора**).

Синтаксис:

```
string GenerateNextClassifierKey (object userSession; string parentKey; Int32 objType)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

parentKey

Значение ключа родительского элемента классификатора.

objType

Тип объектов, для объекта которого генерируется данный ключ, если передается Consts.UnknownObjectId, будет генерировать в разрезе всех объектов в базе данных, у которых есть такой атрибут.

Возвращаемое значение:

Следующее доступное значение для ключа классификатора.

GenerateNextTopLevelKey

Получить следующее доступное значение для ключа классификатора (каталога Imbase либо другого объекта корневого уровня с атрибутом "ключ папки классификатора").

Синтаксис:

```
string GenerateNextTopLevelKey (object userSession)
```

```
string GenerateNextTopLevelKey (object userSession; Int32 objType)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

objType

Идентификатор типа объекта.

Возвращаемое значение:

Новый ключ классификатора верхнего уровня для указанного типа объекта.

GetClassifierForObject

Получить ID папки классификатора, в которую включен объект, и 0, если никуда не включен.

Синтаксис:

```
Int64 GetClassifierForObject (object userSession; Int64 ID)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

ID

ID объекта (F_ID).

Возвращаемое значение:

ID папки классификатора (F_FOLDER_ID).

GetClassifierForObjType

Получить из кэша массив с ID классификаторов, у которых атрибут **Глобальные идентификатор типов объектов** равен передаваемому типу объекта, кроме того все, кому не назначен. Если передать – 1, возвратит все классификаторы.

Синтаксис:

```
Int64 GetClassifierForObjType (object userSession; Int32 objType)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

objType

Тип объектов.

Возвращаемое значение:

массив с ID классификаторов.

GetConditionStructures

Получение массива условий выборки по указанному идентификатору (если есть в кэше - берется из кэша).

Синтаксис:

```
Intermech.Kernel.Search.ConditionStructure[] GetConditionStructures (object userSession; Int64 selectionID)
```

```
Intermech.Kernel.Search.ConditionStructure[] GetConditionStructures (object userSession; Int64 selectionID; Int64 objectID)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

selectionID

Идентификатор выборки.

objectID

Идентификатор для подстановки в условия.

Возвращаемое значение:

Массив элементов ConditionStructure.

GetObjectClassifier

Получить интерфейс на классификатор объектов.

Синтаксис:

```
Intermech.Interfaces.IObjectClassifier GetObjectClassifier (object userSessionGuid; Int64 classifierID)
```

Параметры:

userSessionGuid

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

classifierID

Идентификатор версии объекта классификатора.

Возвращаемое значение:

Обработчик объекта-классификатора.

GetObjectTypesForClassifier

Получить список идентификаторов типов объектов, которым назначен указанный классификатор.

Синтаксис:

```
Int32[] GetObjectTypesForClassifier (object userSession; Int64 classifierID)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

classifierID

Идентификатор классификатора.

Возвращаемое значение:

Список идентификаторов типов объектов, которым назначен указанный классификатор.

GetShowInternalFolders

Получение значения флага, определяющего нужно ли показывать состав вложенных папок классификаторов.

Синтаксис:

```
bool GetShowInternalFolders ()
```

Возвращаемое значение:

Значение флага.

IncludeObjects

Добавление объектов к классификатору (ручной выборке). При неудаче возвращает ID объектов, которые не удалось классифицировать.

Синтаксис:

```
void IncludeObjects (object userSessionGuid; System.Guid selectionGuid; Int64[] objectIDs)
```

```
void IncludeObjects (object userSession; Int64 selectionID; Int64[] objectIDs)
```

Параметры:

userSessionGuid

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

selectionGuid

Глобальный идентификатор классификатора (ручной выборки).

selectionID

Идентификатор классификатора (ручной выборки).

objectIDs

Массив идентификаторов объектов, которые надо добавить.

IsEnabledConditionStructure

Проверить, включено ли условие выборки с индексом conditionIndex.

Синтаксис:

```
bool IsEnabledConditionStructure (Int64 selectionID; Int32 conditionIndex)
```

Параметры:

selectionID

ID выборки.

conditionIndex

Индекс условия в массиве условий выборки.

Возвращаемое значение:

true – указанное условие включено.

LoadClassifierToObjTypeCache

Загрузка данных в кэш.

Синтаксис:

```
void LoadClassifierToObjTypeCache ()
```

SetConditionStructures

Установка нового массива условий для выборки по указанному идентификатору (в базе и в кэше).

Синтаксис:

```
bool SetConditionStructures (object userSession; Int64 selectionID;  
Intermech.Kernel.Search.ConditionStructure[] conditionStructures)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

selectionID

Идентификатор выборки.

conditionStructures

Массив элементов ConditionStructure.

Возвращаемое значение:

true если операция прошла успешно, иначе false.

SetShowInternalFolders

Установка значения флага, определяющего нужно ли показывать состав вложенных папок классификаторов.

Синтаксис:

```
void SetShowInternalFolders (bool newValue)
```

Параметры:

newValue

новое значение флага.

UpdateCashe

Обновление всех имеющихся в кэше условий выборки.

Синтаксис:

```
void UpdateCashe (object userSession)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

UpdateCashe

Обновление в кэше условия для выборки с указанным идентификатором (если его нет в кэше, то будет туда добавлено).

Синтаксис:

```
void UpdateCashe (object userSession; Int64 selectionID)
```

Параметры:

userSession

Пользовательская сессия. В реализации службы выборок на стороне сервера предполагается, что данный параметр типа Guid. Для клиентской службы реализована обработка параметра как IUserSession. Т.е. при обращении к методам службы на сервере нужно передавать Guid пользовательской сессии, а при вызове методов службы на клиенте надо передать ссылку на интерфейс IUserSession.

selectionID

Идентификатор выборки.

7.3 Чтение и сохранение настроек

7.3.1 Интерфейс IDBConfigurations

Интерфейс для чтения и записи настроек пользователей и системы IPS Development Kit.

7.3.1.1 Пространство имен

Intermech.Interfaces

7.3.1.2 Методы интерфейса:

ClearSection

Очищает содержимое секции SectionID модуля ModuleName.

Синтаксис:


```
void ClearSection (string ModuleName; string SectionID; Int64 userID)
```

Параметры:

ModuleName

Имя модуля.

SectionID

Имя секции.

userID

Идентификатор пользователя. Если равен 0, то удаляется секция общих для всех пользователей настроек.

GetConfigAttribute

Возвращает атрибут, в котором хранятся конфигурационные файлы текущего пользователя. Поле Index атрибута устанавливается в нужную позицию, соответствующую имени файла с данными data_name. Если такого файла среди значений атрибута нет, то он добавляется к списку значений, а поле Index устанавливается в соответствующую этому файлу позицию.

Синтаксис:

```
Intermech.Interfaces.IDBAttribute GetConfigAttribute (string data_name)
```

Параметры:

data_name

Имя конфигурационного файла.

Возвращаемое значение:

Атрибут, в котором хранится указанный конфигурационный файл.

LoadConfigData

Считывает конфигурационный файл из настроек текущего пользователя. Если такого файла в настройках нет, то он создается нулевой длины.

Синтаксис:

```
void LoadConfigData (string config_name; Intermech.Interfaces.BlobInformation config_info; byte[] config_file)
```

Параметры:

config_name

Имя конфигурационного файла.

config_info

Описание конфигурационного файла.

config_file

Конфигурационный файл-результат.

LoadConfigData

Считывает конфигурационный файл с именем config_name. config_info - информация о файле, config_file - тело файла. Файл читается из настроек пользователя userID. Для общих конфигураций следует задавать userID = 0.

Синтаксис:

```
void LoadConfigData (string config_name; Intermech.Interfaces.BlobInformation config_info; byte[] config_file; Int64 userID)
```

Параметры:

config_name

Имя конфигурационного файла.

config_info

Описание конфигурационного файла.

config_file

Конфигурационный файл.

userID

Идентификатор объекта пользователя.

ReadBool

Прочитать логический параметр.

Синтаксис:

```
bool ReadBool (string ModuleName; string SectionID; string ParamName; bool DefaultValue; Intermech.DBConfigMode configMode)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

DefaultValue

Значение параметра по умолчанию.

configMode

Метод чтения параметра.

Возвращаемое значение:

Значение логического параметра или значение по умолчанию.

ReadDateTime

Прочитать дату.

Синтаксис:

```
System.DateTime ReadDateTime (string ModuleName; string SectionID; string ParamName; System.DateTime  
DefaultValue; Intermech.DBConfigMode configMode)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

DefaultValue

Значение параметра по умолчанию.

configMode

Метод чтения параметра.

Возвращаемое значение:

Значение даты или значение по умолчанию.

ReadDouble

Прочитать вещественный параметр.

Синтаксис:

```
float64 ReadDouble (string ModuleName; string SectionID; string ParamName;  
float64 DefaultValue; Intermech.DBConfigMode configMode)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

DefaultValue

Значение параметра по умолчанию.

configMode

Метод чтения параметра.

Возвращаемое значение:

Значение вещественного параметра или значение по умолчанию.

ReadInteger

Прочитать целочисленный параметр.

Синтаксис:

```
Int64 ReadInteger (string ModuleName; string SectionID; string ParamName;  
Int64 DefaultValue; Intermech.DBConfigMode configMode)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

DefaultValue

Значение параметра по умолчанию.

configMode

Метод чтения параметра.

Возвращаемое значение:

Значение целочисленного параметра или значение по умолчанию.

ReadSection

Прочитать содержимое секции SectionID для пользователя userID.

Синтаксис:

```
System.Data.DataTable ReadSection (string ModuleName; string SectionID; Int64 userID)
```

Параметры:

ModuleName

Имя модуля.

SectionID

Имя секции.

userID

Идентификатор пользователя. Если равен 0, то читается секция общих для всех пользователей настроек.

Возвращаемое значение:

Возвращаемая таблица содержит список названий параметров F_PARAM_NAME и их значений F_VALUE.

ReadString

Прочитать строковый параметр.

Синтаксис:

```
string ReadString (string ModuleName; string SectionID; string ParamName;  
string DefaultValue; Intermech.DBConfigMode configMode)
```

Параметры:

ModuleName

Имя модуля, которому принадлежит параметр (до 20 символов).

SectionID

Имя секции, в которой записан параметр (до 32 символов).

ParamName

Имя параметра (до 32 символов).

DefaultValue

Значение параметра (до 450 символов).

configMode

Метод чтения параметра (DBConfigMode.UserOnly – только из настроек текущего пользователя, DBConfigMode.GlobalOnly – только из общих настроек, DBConfigMode.UserAndGlobal – сначала производит поиск настройки у текущего пользователя, а в случае ее отсутствия – в общих настройках, DBConfigMode.GlobalAndUser – ищет сперва глобальную настройку, а потом пользовательскую).

Возвращаемое значение:

Значение строкового параметра или значение по умолчанию.

WriteBool

Записать логический параметр.

Синтаксис:

```
Int32 WriteBool (string ModuleName; string SectionID; string ParamName; bool Value)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

WriteBool

Записать логический параметр в конфигурацию указанного пользователя.

Синтаксис:

```
Int32 WriteBool (string ModuleName; string SectionID; string ParamName; bool Value;  
Int64 userID)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

userID

Идентификатор объекта пользователя.

WriteConfigData

Записывает конфигурационный файл config_file с именем config_info.FileName в конфигурационные данные текущего пользователя. Если такой файл уже существует, то он перезаписывается.

Синтаксис:

```
void WriteConfigData (Intermech.Interfaces.BlobInformation config_info; byte[] config_file)
```

Параметры:

config_info

Описание конфигурационного файла.

config_file

Конфигурационный файл.

WriteConfigData

Записывает конфигурационный файл config_file с именем config_info.FileName в конфигурационные данные пользователя userID. Для общих конфигураций следует задавать userID = 0.

Синтаксис:

```
void WriteConfigData (Intermech.Interfaces.BlobInformation config_info; byte[] config_file; Int64  
userID)
```

Параметры:

config_info

Описание конфигурационного файла.

config_file

Конфигурационный файл.

userID

Идентификатор объекта пользователя.

WriteDateTime

Записать дату.

Синтаксис:

```
Int32 WriteDateTime (string ModuleName; string SectionID; string ParamName; System.DateTime Value)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

WriteDateTime

Записать дату в конфигурацию указанного пользователя.

Синтаксис:

```
Int32 WriteDateTime (string ModuleName; string SectionID; string ParamName; System.DateTime Value; Int64 userID)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

userID

Идентификатор объекта пользователя.

WriteDouble

Записать вещественный параметр.

Синтаксис:

```
Int32 WriteDouble (string ModuleName; string SectionID; string ParamName; float64 Value)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

WriteDouble

Записать вещественный параметр в конфигурацию указанного пользователя.

Синтаксис:

```
Int32 WriteDouble (string ModuleName; string SectionID; string ParamName; float64 Value; Int64 userID)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

userID

Идентификатор объекта пользователя.

WriteInteger

Записать целочисленный параметр.

Синтаксис:

```
Int32 WriteInteger (string ModuleName; string SectionID; string ParamName; Int64 Value)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

WriteInteger

Записать целочисленный параметр в конфигурацию указанного пользователя.

Синтаксис:

```
Int32 WriteInteger (string ModuleName; string SectionID; string ParamName; Int64 Value;  
Int64 userID)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

userID

Идентификатор объекта пользователя.

WriteSection

Записывает содержимое секции SectionID модуля ModuleName. table содержит строковые поля с именами (колонок 0) и значениями параметров (колонок 1).

Синтаксис:

```
void WriteSection (string ModuleName; string SectionID; System.Data.DataTable table;  
Int64 userID)
```

Параметры:

ModuleName

Имя модуля.

SectionID

Имя секции.

table

Таблица с именами (колонок 0) и значениями параметров (колонок 1).

userID

Идентификатор пользователя. Если равен 0, то удаляется секция общих для всех пользователей настроек.

WriteString

Записать строковый параметр.

Синтаксис:

```
Int32 WriteString (string ModuleName; string SectionID; string ParamName; string Value)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

WriteString

Записать строковый параметр в конфигурацию указанного пользователя.

Синтаксис:

```
Int32 WriteString (string ModuleName; string SectionID; string ParamName; string Value;  
Int64 userID)
```

Параметры:

ModuleName

Имя модуля, который сохраняет конфигурации.

SectionID

Имя секции.

ParamName

Имя параметра.

Value

Записываемое значение.

userID

Идентификатор объекта пользователя. Если идентификатор равен 0, то читается секция общих для всех пользователей настроек. Для записи общих настроек необходимы права административной роли.

7.3.2 Перечислитель DBConfigMode

Методы чтения (записи) конфигураций.

7.3.2.1 Пространство имен

Intermech

7.3.2.2 Значения перечислителя:

GlobalAndUser = 0x00000003

Читать сначала глобальные настройки, а если их нет, то пользовательские.

GlobalOnly = 0x00000001

Читать (писать) только глобальные настройки.

UserAndGlobal = 0x00000002

Читать сначала пользовательские настройки, а если их нет, то глобальные.

UserOnly = 0x00000000

Читать (писать) только пользовательские настройки.

7.3.3 Интерфейс IContainerService

Интерфейс на сервис работы с контейнерами атрибутов.

7.3.3.1 Пространство имен

Intermech.Interfaces

7.3.3.2 Методы интерфейса:

DeleteContainerForLCLevel

Удалить контейнер атрибутов для уровня продвижения.

Синтаксис:

```
void DeleteContainerForLCLevel (object session; System.Guid LCLevelGuid)
```

Параметры:

session

Пользовательская сессия.

LCLevelGuid

guid уровня продвижения.

DeleteContainerForLCStep

Удалить контейнер атрибутов для шага ЖЦ.

Синтаксис:

```
void DeleteContainerForLCStep (object session; System.Guid LCStepGuid)
```

Параметры:

session

Пользовательская сессия.

LCStepGuid

guid уровня продвижения.

DeleteContainerForLCStepObjectType

Удалить контейнер атрибутов для типа объектов и шага жц.

Синтаксис:

```
void DeleteContainerForLCStepObjectType (object session; System.Guid LCStepGuid; System.Guid  
ObjectTypeGuid)
```

Параметры:

session

Пользовательская сессия.

LCStepGuid

guid шага ЖЦ.

ObjectTypeGuid

guid типа объекта.

DeleteContainerForObjectType

Удалить контейнер атрибутов для типа объектов.

Синтаксис:

```
void DeleteContainerForObjectType (object session; System.Guid objectTypeGuid)
```

Параметры:

session

Пользовательская сессия.

objectTypeGuid

guid типа объекта.

GetContainerForLCLevel

Получить контейнер атрибутов для уровня продвижения.

Синтаксис:

```
Intermech.Interfaces.IDBObject GetContainerForLCLevel (object session;  
System.Guid LCLevelGuid)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCLevel (object session;  
System.Guid LCLevelGuid; bool createIfNotExist)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCLevel (object session; Int32 LCLevelID)
```

Параметры:

session

Пользовательская сессия.

LCLevelGuid

GUID уровня продвижения.

LCLevelID

ID уровня продвижения.

createIfNotExist

Создавать новый контейнер при отсутствии его для данного уровня продвижения.

Возвращаемое значение:

Контейнер атрибутов для указанного уровня продвижения.

GetContainerForLCStep

Получить контейнер атрибутов для шага жизненного цикла.

Синтаксис:

```
Intermech.Interfaces.IDBObject GetContainerForLCStep (object session; System.Guid LCStepGuid)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCStep (object session;  
System.Guid LCStepGuid; bool createIfNotExist)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCStep (object session; Int32 LCStepID)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCStep (object session; Int32 LCStepID; bool  
createIfNotExist)
```

Параметры:

session

Пользовательская сессия.

LCStepGuid

GUID шага ЖЦ.

LCStepID

ID шага ЖЦ.

createIfNotExist

Создавать новый контейнер при отсутствии его для данного шага ЖЦ.

Возвращаемое значение:

Контейнер атрибутов для указанного шага ЖЦ.

GetContainerForLCStepObjectType

Получить контейнер атрибутов для шага ЖЦ и типа объекта.

Синтаксис:

```
Intermech.Interfaces.IDBObject GetContainerForLCStepObjectType (object session; System.Guid LCStep;  
System.Guid ObjectTypeGuid)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCStepObjectType (object session; System.Guid LCStep;  
System.Guid ObjectTypeGuid; bool createIfNotExist)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCStepObjectType (object session; Int32 LCStepID; Int32  
ObjectTypeID)
```

```
Intermech.Interfaces.IDBObject GetContainerForLCStepObjectType (object session; Int32 LCStepID; Int32  
ObjectTypeID; bool createIfNotExist)
```

Параметры:

session

Пользовательская сессия.

LCStep

guid шага ЖЦ.

ObjectTypeGuid

guid типа объекта.

ObjectTypeID

id типа объекта.

createIfNotExist

Создавать новый контейнер при его отсутствии.

Возвращаемое значение:

Контейнер атрибутов для указанного шага ЖЦ и типа объектов.

GetContainerForObjectType

Получить контейнер атрибутов для типа объекта.

Синтаксис:

```
Intermech.Interfaces.IDBObject GetContainerForObjectType (object session;  
System.Guid objectTypeGuid)
```

Параметры:

session

Пользовательская сессия.

objectTypeGuid

GUID типа объекта.

Возвращаемое значение:

Контейнер атрибутов для указанного типа объекта.

GetContainerForObjectType

Получить контейнер атрибутов для типа объекта.

Синтаксис:

```
Intermech.Interfaces.IDBObject GetContainerForObjectType (object session;  
System.Guid objectTypeGuid; bool createIfNotExist)
```

```
Intermech.Interfaces.IDBObject GetContainerForObjectType (object session;  
Int32 objectTypeID)
```

```
Intermech.Interfaces.IDBObject GetContainerForObjectType (object session;  
Int32 objectTypeID; bool createIfNotExist)
```

Параметры:

session
Пользовательская сессия.
objectTypeGuid
GUID типа объекта.
objectTypeID
ID типа объекта.
createIfNotExist
Создавать новый контейнер при отсутствии его для данного типа объекта.
Возвращаемое значение:
Контейнер атрибутов для указанного типа объекта.

7.3.4 Примеры

Для конкретного типа объекта можно получить контейнер атрибутов:

Получаем ссылку на интерфейс сервиса для работы с контейнером атрибутов

```
using(SessionKeeper keeper = new SessionKeeper())
{
    IUserSession session = keeper.Session;
    IContainerService containerService =
    (IContainerService)session.GetCustomService(typeof(IContainerService));
}
```

Интерфейс поддерживает пока только 1 метод в 4 вариантах:

Получить контейнер для типа объекта

По GUID типа объекта

Если передать **create = true**, то при неудачном поиске в уже существующих контейнерах – будет создан новый контейнер атрибутов для переданного типа объектов.

Второй метод – тот же поиск, только без создания.

IDBObject GetContainerForObjectType(IUserSession session, Guid objectTypeGuid, bool create);

IDBObject GetContainerForObjectType(IUserSession session, Guid objectTypeGuid);

По ID типа объекта. Все тоже самое.

IDBObject GetContainerForObjectType(IUserSession session, int objectTypeID, bool create);

IDBObject GetContainerForObjectType(IUserSession session, int objectTypeID);

Если контейнер не найден и **create = false** или **2 и 4 метод** - получим **null**.

7.4 Вспомогательные службы

7.4.1 Интерфейс ICacheDataset

Служба кэширования системной информации на сервере приложений. Часть методов интерфейса используется ядром системы для обновления кэша, поэтому не рекомендуется использовать функции интерфейса, не описанные в «Руководстве программиста».

7.4.1.1 Пространство имен

Intermech.Interfaces.Server

7.4.1.2 Свойства интерфейса:

System.DateTime ModifyDate { get; }

Дата и время последней модификации кэша таблиц метаданных (в UTC).

7.4.1.3 Методы интерфейса:

GetAttributeOptions

Возвращает опции атрибута (в т. ч. применительно к типу объектов и связей). Возвращает AttributeOptions.None в случае, если указанного атрибута в базе не существует (или он не назначен указанному типу объектов или связей).

Синтаксис:

Intermech.AttributeOptions GetAttributeOptions (Int32 attributeID; Int32 objectTypeID;
Int32 relationTypeID)

Параметры:

attributeID

Идентификатор атрибута.

objectTypeID

Идентификатор типа объектов, применительно к которому нужно получить опции атрибута (или -1, если не требуется получать опции атрибута применительно к типу объектов).

relationTypeID

Идентификатор типа связей, применительно к которому нужно получить опции атрибута (или -1, если не требуется получать опции атрибута применительно к типу связей).

Возвращаемое значение:

Опции атрибута.

GetFilePrototype

Функция возвращает массив идентификаторов объектов, которые хранят прототипы файлов для атрибута attributeID у типа объектов objectTypeID и пользователя userID. Возвращает null если ничего не найдено.

Синтаксис:

Int64 GetFilePrototype (Int32 attributeID; Int32 objectTypeID; Int64 userID)

Параметры:

attributeID

Идентификатор атрибута.

objectTypeID

Идентификатор типа объектов.

userID

Идентификатор пользователя, для которого ищется прототип.

Возвращаемое значение:

Массив идентификаторов объектов, которые хранят прототипы файлов для атрибута attributeID.

GetObjectTypeGuid

Возвращает GUID для типа объектов номер objectTypeID. Если throwIfNotFound == false, то при отсутствии типа возвращает Guid.Empty.

Синтаксис:

System.Guid GetObjectTypeGuid (Int32 objectTypeID; bool throwIfNotFound)

Параметры:

objectTypeID

Локальный идентификатор типа объектов.

throwIfNotFound

Если значение параметра throwIfNotFound равно false, то при отсутствии типа с указанным идентификатором функция возвращает Guid.Empty.

Возвращаемое значение:

GUID для типа объектов номер objectTypeID.

GetObjectTypeParentID

Возвращает идентификатор родительского типа объектов для типа objectTypeID (или -1 если родителя нет).

Синтаксис:

Int32 GetObjectTypeParentID (Int32 objectTypeID)

Параметры:

objectTypeID

Дочерний тип объекта.

Возвращаемое значение:

Идентификатор родительского типа объектов.

GetOptimizationMode

Возвращает режим оптимизации атрибута (в т. ч. применительно к типу объектов и связей). Возвращает значение OptimizationModes.NotFound в случае, если атрибут не найден у указанных типов.

Синтаксис:

Intermech.OptimizationModes GetOptimizationMode (Intermech.Interfaces.Server.Attribute4ID attrStruct)

Intermech.OptimizationModes GetOptimizationMode (Int32 attributeID)

Intermech.OptimizationModes GetOptimizationMode (Int32 attributeID; Int32 objectTypeID; Int32 relationTypeID)

Параметры:

attrStruct

Структура, предназначенная для хранения набора идентификаторов атрибута применительно к типам объектов или связей.

attributeID

Идентификатор атрибута.

objectTypeID

Идентификатор типа объектов (-1 если атрибут не рассматривается применительно к типу объектов).

relationTypeID

Идентификатор типа связей (-1 если атрибут не рассматривается применительно к типу связей).

Возвращаемое значение:

Режим оптимизации указанного атрибута.

GetTable

Получить ссылку на таблицу tableName из кэша метаданных.

Синтаксис:

System.Data.DataTable GetTable (string tableName)

Параметры:

tableName

Имя таблицы.

Возвращаемое значение:

Ссылка на таблицу, хранящуюся в кэше метаданных.

LoadFilePrototypes

Процедура перечитывает хэш-таблицу со списком прототипов для файловых атрибутов. Если objectTypeID != -1, то перечитываются только прототипы для типа объектов objectTypeID.

Синтаксис:

void LoadFilePrototypes (Intermech.Interfaces.IUserSession session; Int32 objectTypeID)

Параметры:

session

Сессия.

objectTypeID

Идентификатор типа объекта или -1.

LoadTables

Перечитывает из СУБД все таблицы в кэш метаданных.

Синтаксис:

void LoadTables (Intermech.Interfaces.Server.IDbManager db)

Параметры:

db

Ссылка на интерфейс для работы с базой данных.

ReloadTables

Перечитать в кэше таблицы из заданного списка.

Синтаксис:

void ReloadTables (Intermech.Interfaces.IUserSession uSession; Intermech.Interfaces.Server.IDbManager db; string tablesList)

Параметры:

uSession

Сессия.

db

Ссылка на интерфейс для работы с базой данных.

tablesList

Список названий таблиц метаданных, которые требуется перечитать в кэш.

UpdateObjectInfo

Добавляет или обновляет в кэше информацию об объекте info.ObjectID.

Синтаксис:

```
void UpdateObjectInfo (Intermech.Interfaces.QuickObjectInfo info)
```

Параметры:

info

Структура с краткой информацией о версии объекта.

7.4.2 Класс Attribute4ID

Класс, предназначенный для хранения набора идентификаторов атрибута применительно к типам объектов или связей.

7.4.2.1 Пространство имен

Intermech.Interfaces.Server

7.4.2.2 Поля класса:

Int32 AttributeID

Идентификатор типа атрибута.

Int32 ObjectTypeID

Идентификатор типа объекта.

Int32 RelationTypeID

Идентификатор типа связи.

7.4.2.3 Методы класса:

Attribute4ID (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void Attribute4ID (Int32 attributeID)
```

```
void Attribute4ID (Int32 attributeID; Int32 objectTypeID; Int32 relationTypeID)
```

Параметры:

attributeID

Идентификатор типа атрибута.

objectTypeID

Идентификатор типа объекта.

relationTypeID

Идентификатор типа связи.

Equals

Сравнить экземпляр класса с другим объектом.

Синтаксис:

```
bool Equals (object obj)
```

Параметры:

obj

Объект для сравнения.

Возвращаемое значение:

Объекты равны.

GetHashCode

Вернуть 32-битный хэш-код экземпляра класса.

Синтаксис:

```
Int32 GetHashCode ()
```

Возвращаемое значение:

32-битный хэш-код экземпляра класса.

7.4.3 Интерфейс IBrowseFolder

Интерфейс для просмотра файловой системы сервера. Доступен на стороне клиента через интерфейс портфеля Intermech.Interfaces.Briefcase.IServerBriefcase.

7.4.3.1 Пространство имен

Intermech.Interfaces

7.4.3.2 Свойства интерфейса:

```
string DrivesList { get; }
```

Список дисков, записанный в виде C:\.

7.4.3.3 Методы интерфейса:

GetFolders

Получить список папок, входящих в указанную родительскую папку.

Синтаксис:

```
string GetFolders (string parentPath)
```

Параметры:

parentPath

Родительская папка.

Возвращаемое значение:

Список папок, входящих в указанную родительскую папку.

GetFreeSpace

Возвращает количество свободного места на диске в байтах, на котором находится папка folderPath.

Синтаксис:

```
Int64 GetFreeSpace (string folderPath)
```

Параметры:

folderPath

Полное имя папки.

Возвращаемое значение:

Возвращает количество свободного места на диске (в байтах).

RefreshDrivesList

Обновляет список дисков сервера.

Синтаксис:

```
void RefreshDrivesList ()
```

7.5 Анализ списка удаляемых объектов

Удаление объектов из Навигатора выполняется следующим образом:

- формируется список удаляемых объектов - **DeletingObjects**;
- при необходимости данный список обрабатывается серверной службой **IObjectsDeleteAnalyzerService**, которая позволяет отыскать зависимые от удаляемых объектов связи и другие объекты, которые тоже может потребоваться удалить;
- окончательно сформированный список **DeletingObjects** передается серверной службе **IObjectsDeleteService**, которая выполняет удаление связей и объектов, указанных в данном списке.

Рассмотрим подробно каждый элемент, используемый в процессе удаления объектов и связей.

7.5.1 Интерфейс IObjectsDeleteAnalyzerService

Интерфейс серверной службы анализаторов списка удаляемых объектов.

7.5.1.1 Пространство имен

Intermech.Interfaces

7.5.1.2 Методы интерфейса:

Analyze

Начать выполнение анализа удаляемых объектов, при необходимости добавить в граф дополнительные идентификаторы версий объектов, которые тоже требуется удалить. На верхнем уровне - первоначальный список удаляемых версий объектов. Служба опрашивает все зарегистрированные в ней анализаторы для

выполнения данного анализа. Вся работа выполняется на сервере в отдельном потоке, в рамках задания по анализу.

Синтаксис

```
System.Guid Analyze (System.Guid sessionGuid;  
Intermech.Interfaces.DeletingObjects deletingObjects;  
Intermech.Interfaces.DeleteAnalyzerOptions options)
```

Параметры:

sessionGuid

идентификатор сессии, в рамках которой выполняется анализ.

deletingObjects

Список удаляемых версий объектов.

options

Параметры.

Возвращаемое значение:

Уникальный идентификатор задания, в рамках которого выполняется анализ. Значение Guid.Empty означает невозможность начать анализ.

CancelJob

Прервать указанное задание.

Синтаксис:

```
bool CancelJob (System.Guid jobID)
```

Параметры:

jobID

Задание.

Возвращаемое значение:

true, если задание было найдено и остановлено.

LoadDescriptions

Заполнить описания в указанных объектах.

Синтаксис

```
Intermech.Interfaces.DeletingObjects LoadDescriptions (System.Guid sessionGuid;  
Intermech.Interfaces.DeletingObjects deletingObjects)
```

Параметры:

sessionGuid

идентификатор сессии, в рамках которой выполняется анализ.

deletingObjects

Список удаляемых версий объектов.

Возвращаемое значение:

Список удаляемых объектов, у которых заполнены все поля.

QueryJobStatus

Запросить статус указанного задания на сервере. Если задание успешно или ошибочно завершено, вернётся полный пакет со статусом задания, а само задание будет удалено на серверной стороне вместе со своим потоком.

Синтаксис

```
Intermech.Interfaces.DeleteAnalyzerJobStatus QueryJobStatus (System.Guid jobID)
```

Параметры:

jobID

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

RegisterAnalyzer

Выполнить регистрацию анализатора в службе.

Синтаксис

```
bool RegisterAnalyzer (Intermech.Interfaces.IObjectsDeleteAnalyzer analyzer)
```

Параметры:

analyzer

Анализатор.

Возвращаемое значение:

true, если регистрация выполнена успешно.

UnregisterAnalyzer

Выполнить удаление анализатора из службы.

Синтаксис

bool UnregisterAnalyzer (Intermech.Interfaces.IObjectsDeleteAnalyzer analyzer)

Параметры:

analyzer

Анализатор.

Возвращаемое значение:

true, если удаление выполнено успешно.

UnregisterAnalyzer

Выполнить удаление анализатора по его Guid из службы.

Синтаксис

bool UnregisterAnalyzer (System.Guid analyzerGuid)

Параметры:

analyzerGuid

Guid анализатора.

Возвращаемое значение:

true, если удаление выполнено успешно.

7.5.2 Класс DeletingObject

Описание удаляемого объекта.

7.5.2.1 Пространство имен

Intermech.Interfaces

7.5.2.2 Реализуемые интерфейсы:

System.ICloneable

7.5.2.3 Свойства класса:

Intermech.Interfaces.DeletingObject this [Int32 index]

Дочернее описание удаляемого объекта с указанным индексом.

Параметры свойства:

index

Индекс удаляемого объекта.

bool BaseVersion { get; set; }

Является ли версия базовой.

string Caption { get; set; }

Заголовок удаляемого объекта.

Int64 ChkOutByID { get; set; }

Кем объект взят на изменение.

Int32 Count { get; }

Количество дочерних описаний удаляемых объектов.

Int64 ID { get; set; }

Идентификатор удаляемого объекта.

Intermech.Interfaces.DeletingObjects Items { get; }

Коллекция дочерних описаний удаляемых объектов.

Int32 LCStepID { get; set; }

Шаг жизненного цикла.

Int64 ObjectID { get; set; }

Идентификатор версии удаляемого объекта (уникальный в пределах всей коллекции).


```

Int32 ObjectType { get; set; }
Идентификатор типа удаляемого объекта.
Int64 OwnerID { get; set; }
Идентификатор владельца объекта.
Intermech.Interfaces.DeletingObjects Parent { get; }
Родительская коллекция удаляемых объектов.
System.Collections.Generic.List<Int64> PrjLinkIDs { get; }
Идентификаторы связей (если объект удаляется из состава).
string RemoveNote { get; set; }
Причина удаления объекта.
bool RemoveObject { get; set; }
Требуется ли удалять указанный объект.
Int64 VersionNo { get; set; }
Номер версии объекта.

```

7.5.2.4 Методы класса:

DeletingObject (конструктор)

Создать описание удаляемого объекта.

Синтаксис:

```

void DeletingObject (Intermech.Interfaces.DeletingObjects parent; Int64 id; Int64 objectID; bool
removeObject)
void DeletingObject (Intermech.Interfaces.DeletingObjects parent; Int64 id; Int64 objectID; bool
removeObject; string removeNote)
void DeletingObject (Intermech.Interfaces.DeletingObjects parent; Int64 id; Int64 objectID; bool
removeObject; Int32 objectType; string caption; Int64 ownerID; Int64 chkOutByID; Int32 lcStepID; string
removeNote)
void DeletingObject (Intermech.Interfaces.DeletingObjects parent; Int64 id; Int64 objectID; bool
removeObject; Int32 objectType; string caption; Int64 ownerID; Int64 chkOutByID; Int32 lcStepID; Int64
prjLinkId; Int64 versionNo; bool baseVersion; string removeNote)

```

Параметры:

parent

Родительская коллекция удаляемых объектов.

id

Идентификатор удаляемого объекта.

objectID

Идентификатор версии удаляемого объекта (уникальный в пределах всей коллекции).

removeObject

Требуется ли удалять указанный объект.

objectType

Идентификатор типа удаляемого объекта.

caption

Заголовок удаляемого объекта.

ownerID

Идентификатор владельца объекта.

chkOutByID

Кем объект взят на изменение.

lcStepID

Шаг жизненного цикла.

prjLinkId

Идентификатор связи (если объект удаляется из состава).

versionNo

Номер версии.

baseVersion

Является ли версия базовой.

removeNote

Причина удаления объекта.

Clone

Создать копию экземпляра класса.

Синтаксис:

`object Clone ()`

Возвращаемое значение:

Копия экземпляра класса.

Equals

Сравнить экземпляр класса с указанным объектом.

Синтаксис:

`bool Equals (object obj)`

Параметры:

`obj`

Объект для сравнения.

Возвращаемое значение:

true - объекты полностью идентичны

FindDeletingObject

Отыскать в коллекции описание удаляемого объекта с указанным идентификатором. Поиск также будет проходить в дочерних коллекциях.

Синтаксис:

`Intermech.Interfaces.DeletingObject FindDeletingObject (Int64 objectID)`

Параметры:

`objectID`

Уникальный в пределах всей коллекции идентификатор версии объекта.

Возвращаемое значение:

`null`, если описание удаляемого объекта не найдено.

GetHashCode

Вернуть 32-битный хэш-код экземпляра класса.

Синтаксис:

`Int32 GetHashCode ()`

Возвращаемое значение:

32-битный хэш-код экземпляра класса.

LoadDescription

Загрузить описание объекта из базы данных (если оно не загружено).

Синтаксис:

`void LoadDescription (Intermech.Interfaces.IUserSession session)`

Параметры:

`session`

Сессия.

7.5.3 Класс DeletingObjects

Коллекция описаний удаляемых объектов.

7.5.3.1 Пространство имен

Intermech.Interfaces

7.5.3.2 Реализуемые интерфейсы:

System.ICloneable

7.5.3.3 Свойства класса:

Intermech.Interfaces.DeletingObjects Parent { get; }

Родительская коллекция описаний удаляемых объектов.

7.5.3.4 Методы класса:

DeletingObjects (конструктор)

Синтаксис:

```
void DeletingObjects ()  
void DeletingObjects (Intermech.Interfaces.DeletingObjects parent)
```

Параметры:

parent

Родительская коллекция.

Add

Добавить описание удаляемого объекта в коллекцию.

Синтаксис:

```
Intermech.Interfaces.DeletingObject Add (Int64 id; Int64 objectID; bool removeObject)  
Intermech.Interfaces.DeletingObject Add (Int64 id; Int64 objectID; bool removeObject; string removeNote)  
Intermech.Interfaces.DeletingObject Add (Int64 id; Int64 objectID; bool removeObject; Int32 objectType;  
string caption; Int64 ownerID; Int64 chkOutByID; Int32 lcStepID; string removeNote)  
Intermech.Interfaces.DeletingObject Add (Int64 id; Int64 objectID; bool removeObject; Int32 objectType;  
string caption; Int64 ownerID; Int64 chkOutByID; Int32 lcStepID; Int64 prjLinkID; Int64 versionNo; bool  
baseVersion; string removeNote)
```

Параметры:

id

Идентификатор удаляемого объекта.

objectID

Идентификатор версии удаляемого объекта (уникальный в пределах всей коллекции).

removeObject

Требуется ли удалять указанный объект.

objectType

Идентификатор типа удаляемого объекта.

caption

Заголовок удаляемого объекта.

ownerID

Идентификатор владельца объекта.

chkOutByID

Кем объект взят на изменение.

lcStepID

Шаг жизненного цикла.

prjLinkID

Идентификатор связи (если объект удаляется из состава).

versionNo

Номер версии.

baseVersion

Признак базовой версии.

removeNote

Причина удаления объекта.

Возвращаемое значение:

Найденное или новое описание удаляемого объекта.

Assign

Полное присваивание другого списка описаний удаляемых объектов.

Синтаксис:

```
void Assign (Intermech.Interfaces.DeletingObjects source)
```

Параметры:

source

Источник.

Clone

Создать копию экземпляра класса.

Синтаксис:

```
object Clone ()
```

Возвращаемое значение:

Копия экземпляра класса.

ExtractAbsDeletingObjects

Получение полного списка описаний удаляемых объектов, включая объекты дочерних коллекций. В коллекцию попадают только положительные идентификаторы.

Синтаксис:

```
System.Collections.Generic.List<DeletingObject> ExtractAbsDeletingObjects ()
```

Возвращаемое значение:

Полный список описаний удаляемых объектов, включая объекты дочерних коллекций.

ExtractAbsIDs

Получение полного списка идентификаторов версий удаляемых объектов, включая объекты дочерних коллекций. В коллекцию попадают только положительные идентификаторы, причём только у выделенных для удаления объектов.

Синтаксис:

```
System.Collections.Generic.List<Int64> ExtractAbsIDs ()
```

Возвращаемое значение:

Полный список идентификаторов версий удаляемых объектов, включая объекты дочерних коллекций.

ExtractDeletingObjects

Получение полного списка описаний удаляемых объектов, включая объекты дочерних коллекций.

Синтаксис:

```
System.Collections.Generic.List<DeletingObject> ExtractDeletingObjects ()
```

Возвращаемое значение:

Полный список описаний удаляемых объектов, включая объекты дочерних коллекций.

FindDeletingObject

Отыскать в коллекции описание удаляемого объекта с указанным идентификатором. Поиск также будет проходить в дочерних коллекциях.

Синтаксис:

```
Intermech.Interfaces.DeletingObject FindDeletingObject (Int64 objectID)
```

Параметры:

objectID

Уникальный в пределах коллекции идентификатор версии объекта.

Возвращаемое значение:

null, если описание удаляемого объекта не найдено.

FindDeletingObjectFromRoot

Отыскать в коллекции (начиная с её корневой записи) описание удаляемого объекта с указанным идентификатором. Поиск также будет проходить в дочерних коллекциях.

Синтаксис:

```
Intermech.Interfaces.DeletingObject FindDeletingObjectFromRoot (Int64 objectID)
```

Параметры:

objectID

Уникальный в пределах коллекции идентификатор версии объекта.

Возвращаемое значение:

null, если описание удаляемого объекта не найдено.

FindRootParent

Отыскивает первого, не равного null, "родителя". Если такого нет, вернёт parObject.

Синтаксис:

```
Intermech.Interfaces.DeletingObjects FindRootParent ()
```

Возвращаемое значение:

Родительский узел, у которого Parent = null.

SelectedCount

Подсчитать, сколько описаний удаляемых объектов выделено.

Синтаксис:

Int32 SelectedCount ()

Возвращаемое значение:

Количество выделенных описаний удаляемых объектов.

Sort

Выполнить сортировку коллекции и её дочерних элементов.

Синтаксис:

```
void Sort (System.Collections.Generic.IComparer<DeletingObject> comparer)
```

Параметры:

comparer

Сравнитель.

7.5.4 Интерфейс IObjectsDeleteAnalyzer

Интерфейс анализатора списка удаляемых объектов.

7.5.4.1 Пространство имен

Intermech.Interfaces

7.5.4.2 Свойства интерфейса:

System.Guid Guid { get; }

Уникальный идентификатор анализатора (по данному идентификатору происходит регистрация и удаление анализатора в службе анализаторов).

7.5.4.3 Методы интерфейса:

Analyze

Выполнить анализ удаляемых объектов, при необходимости добавить в граф дополнительные идентификаторы версий объектов, которые тоже требуется удалить. На верхнем уровне - первоначальный список удаляемых версий объектов.

Синтаксис:

```
Int32 Analyze (Intermech.Interfaces.IUserSession session;
```

```
Intermech.Interfaces.DeletingObjects deletingObjects;
```

```
Intermech.Interfaces.DeleteAnalyzerOptions options)
```

Параметры:

session

Сессия, в рамках которой выполняется анализ.

deletingObjects

Список удаляемых версий объектов.

options

Параметры.

Возвращаемое значение:

Количество добавленных к удалению объектов.

7.5.5 Перечислитель DeleteAnalyzerJobProgress

Индикатор выполнения задания.

7.5.5.1 Пространство имен

Intermech.Interfaces

7.5.5.2 Значения перечислителя:

Cancelled = 0xFFFFFFFF

Задание прервано пользователем.

Completed = 0x00000002

Задание успешно завершено.

Error = 0xFFFFFFFFE

Задание прервано из-за ошибки.

NotStarted = 0x00000000

Задание ещё не запущено.

Working = 0x00000001

Задание работает.

7.5.6 Класс DeleteAnalyzerJobStatus

Состояние задания по анализу списка удаляемых объектов.

7.5.6.1 Пространство имен

Intermech.Interfaces

7.5.6.2 Реализуемые интерфейсы:

System.ICloneable

7.5.6.3 Поля класса:

System.Exception Exception

Исключение, которое возникло в процессе удаления объектов/связей.

Intermech.Interfaces.DeletingObjects Items

Список описаний удаляемых объектов. Значение будет заполнено только когда задание будет успешно выполнено.

Int64 Objects

Количество объектов, добавленных для удаления.

Intermech.Interfaces.DeleteAnalyzerJobProgress Progress

Индикатор выполнения задания.

7.5.6.4 Методы класса:

DeleteAnalyzerJobStatus (конструктор)

Создать экземпляр класса.

Синтаксис:

void DeleteAnalyzerJobStatus ()

Cancel

Установить поля класса в "Задание прервано".

Синтаксис:

void Cancel ()

Clone

Создать копию экземпляра класса.

Синтаксис:

object Clone ()

Возвращаемое значение:

Копия экземпляра класса.

Complete

Установить поля класса в "Задание успешно выполнено".

Синтаксис:

void Complete (Int32 objects; Intermech.Interfaces.DeletingObjects items)

Параметры:

objects

Количество добавленных описаний объектов.

items

Коллекция описаний удаляемых объектов.

Error

Установить поля класса в "Задание остановлено из-за ошибки".

Синтаксис:

void Error (System.Exception exception; Intermech.Interfaces.DeletingObjects items)

Параметры:

exception

Возникшее исключение.

items

Коллекция описаний удаляемых объектов.

Start

Установить поля класса в "Задание стартовало".

Синтаксис:

```
void Start ()
```

7.5.7 Класс DeletingObjectComparer

Базовый класс для сравнителей удаляемых объектов. Используется для сортировки удаляемых объектов в списке DeletingObjects. На его основе реализованы описанные далее классы.

7.5.7.1 *Пространство имен*

Intermech.Interfaces

7.5.7.2 *Реализуемые интерфейсы:*

System.Collections.Generic.IComparer<DeletingObject>

7.5.7.3 *Поля класса:*

protected bool ascending

Сортировать по возрастанию.

7.5.7.4 *Методы класса:*

DeletingObjectComparer (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void DeletingObjectComparer (bool ascending)
```

Параметры:

ascending

true - сортировать по возрастанию.

Compare

Сравнить два объекта для удаления.

Синтаксис:

```
Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)
```

Параметры:

x

Первый удаляемый объект.

y

Второй удаляемый объект.

Возвращаемое значение:

-1, 0 или 1, в зависимости от результатов сравнения.

InvertResult

Вернуть результат сравнения в зависимости от значения флажка ascending.

Синтаксис:

```
Int32 InvertResult (Int32 value)
```

Параметры:

value

Оригинальный результат сравнения после вызова метода Compare.

Возвращаемое значение:

Результат сравнения в зависимости от значения флажка ascending.

7.5.8 Класс DeletingObjectComparerCaption

Сравнить два описания удаляемых объектов по их заголовкам.

7.5.8.1 *Пространство имен*

Intermech.Interfaces

7.5.8.2 Реализуемые интерфейсы:

System.Collections.Generic.IComparer<DeletingObject>

7.5.8.3 Методы класса:

DeletingObjectComparerCaption (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void DeletingObjectComparerCaption (bool ascending)
```

Параметры:

ascending

true - сортировать по возрастанию.

Compare

Сравнить два описания удаляемых объектов по заголовкам.

Синтаксис:

```
Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)
```

Параметры:

x

Описание первого удаляемого объекта.

y

Описание второго удаляемого объекта.

Возвращаемое значение:

Результаты сравнения. Класс DeletingObjectComparerObjectID

Сравнить два описания удаляемых объектов по идентификаторов версий объектов.

7.5.8.4 Пространство имен

Intermech.Interfaces

7.5.8.5 Реализуемые интерфейсы:

System.Collections.Generic.IComparer<DeletingObject>

7.5.8.6 Методы класса:

DeletingObjectComparerObjectID (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void DeletingObjectComparerObjectID (bool ascending)
```

Параметры:

ascending

true - сортировать по возрастанию.

Compare

Сравнить два описания удаляемых объектов по идентификаторам версий объектов.

Синтаксис:

```
Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)
```

Параметры:

x

Описание первого удаляемого объекта.

y

Описание второго удаляемого объекта.

Возвращаемое значение:

Результаты сравнения.

7.5.9 Класс DeletingObjectComparerOwnerID

Сравнить два описания удаляемых объектов по их владельцам.

7.5.9.1 Пространство имен

Intermech.Interfaces

7.5.9.2 Реализуемые интерфейсы:

System.Collections.Generic.IComparer<DeletingObjectComparerOwnerID>

7.5.9.3 Методы класса:

DeletingObjectComparerOwnerID (конструктор)

Создать экземпляр класса

Синтаксис:

```
void DeletingObjectComparerOwnerID (bool ascending; System.Collections.Generic.Dictionary<Int64, object> users)
```

Параметры:

ascending

true - сортировать по возрастанию.

users

Кэш имён пользователей. [(Int64)Идентификатор пользователя] => [(object)Имя пользователя (string)].

Compare

Сравнить два описания удаляемых объектов по их владельцам.

Синтаксис:

```
Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)
```

Параметры:

x

Описание первого удаляемого объекта.

y

Описание второго удаляемого объекта.

Возвращаемое значение:

Результаты сравнения.

7.5.10 Класс DeletingObjectComparerChkOutBy

Сравнить два описания удаляемых объектов по пользователям, взявшим объекты на редактирование.

7.5.10.1 Пространство имен

Intermech.Interfaces

7.5.10.2 Реализуемые интерфейсы:

System.Collections.Generic.IComparer<DeletingObject>

7.5.10.3 Методы класса:

DeletingObjectComparerChkOutBy (конструктор)

Создать экземпляр класса

Синтаксис:

```
void DeletingObjectComparerChkOutBy (bool ascending; System.Collections.Generic.Dictionary<Int64, object> users)
```

Параметры:

ascending

true - сортировать по возрастанию

users

Кэш имён пользователей. [(Int64)Идентификатор пользователя] => [(object)Имя пользователя (string)].

Compare

Сравнить два описания удаляемых объектов по пользователям, взявшим объекты на редактирование.

Синтаксис:

```
Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)
```

Параметры:

x

Описание первого удаляемого объекта.

у

Описание второго удаляемого объекта.

Возвращаемое значение:

Результаты сравнения.

7.5.11 Класс `DeletingObjectComparerObjectType`

Сравнить два описания удаляемых объектов по типам их объектов.

7.5.11.1 *Пространство имен*

Intermech.Interfaces

7.5.11.2 *Реализуемые интерфейсы:*

System.Collections.Generic.IComparer<DeletingObject>

7.5.11.3 *Методы класса:*

DeletingObjectComparerObjectType (конструктор)

Создать экземпляр класса

Синтаксис:

```
void DeletingObjectComparerObjectType (bool ascending; System.Collections.Generic.Dictionary<Int64 , object> users; System.Collections.Generic.Dictionary<Int32 , object> types)
```

Параметры:

ascending

true - сортировать по возрастанию.

users

Кэш имён пользователей. [(Int64)Идентификатор пользователя] => [(object)Имя пользователя (string)].

types

Кэш названий типов объектов. [(Int32)Идентификатор типа объекта] => [(object)Имя типа объекта (string)].

Compare

Сравнить два описания удаляемых объектов по типам их объектов.

Синтаксис:

```
Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)
```

Параметры:

x

Описание первого удаляемого объекта.

y

Описание второго удаляемого объекта.

Возвращаемое значение:

Результаты сравнения.

7.5.12 Класс `DeletingObjectComparerLCStep`

Сравнить два описания удаляемых объектов по шагам ЖЦ.

7.5.12.1 *Пространство имен*

Intermech.Interfaces

7.5.12.2 *Реализуемые интерфейсы:*

System.Collections.Generic.IComparer<DeletingObject>

7.5.12.3 *Методы класса:*

DeletingObjectComparerLCStep (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void DeletingObjectComparerLCStep (bool ascending;  
System.Collections.Generic.Dictionary<Int64 , object> users;  
System.Collections.Generic.Dictionary< Int32 , object > lcsteps)
```

Параметры:

ascending

true - сортировать по возрастанию.

users

Кэш имён пользователей. [(Int64)Идентификатор пользователя] => [(object)Имя пользователя (string)].

1csteps

Кэш названий шагов ЖЦ. [(Int32)Идентификатор шага ЖЦ] => [(object)Название шага ЖЦ (string)].

Compare

Сравнить два описания удаляемых объектов по шагам ЖЦ.

Синтаксис:

Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)

Параметры:

x

Описание первого удаляемого объекта.

y

Описание второго удаляемого объекта.

Возвращаемое значение:

Результаты сравнения.

7.5.13 Класс DeletingObjectComparerBaseVersion

Сравнить два описания удаляемых объектов по их принадлежности к базовым версиям.

7.5.13.1 Пространство имен

Intermech.Interfaces

7.5.13.2 Реализуемые интерфейсы:

System.Collections.Generic.IComparer<DeletingObject>

7.5.13.3 Методы класса:

DeletingObjectComparerBaseVersion (конструктор)

Создать экземпляр класса.

Синтаксис:

void DeletingObjectComparerBaseVersion (bool ascending)

Параметры:

ascending

true - сортировать по возрастанию.

Compare

Сравнить два описания удаляемых объектов по принадлежности к базовым версиям.

Синтаксис:

Int32 Compare (Intermech.Interfaces.DeletingObject x; Intermech.Interfaces.DeletingObject y)

Параметры:

x

Описание первого удаляемого объекта.

y

Описание второго удаляемого объекта.

Возвращаемое значение:

Результаты сравнения.

7.5.14 Класс DeleteAnalyzerJob

Задание по анализу списка удаляемых объектов, выполняемое на сервере в отдельном потоке.

7.5.14.1 Пространство имен

Intermech.Kernel

7.5.14.2 Поля класса:

`System.Collections.Generic.Dictionary<Guid, IObjectsDeleteAnalyzer> Analyzers`

Коллекция анализаторов, к которым будет обращаться задание в ходе выполнения анализа.

`System.Guid Guid`

Уникальный идентификатор задания.

`Intermech.Interfaces.DeletingObjects Items`

Список описаний удаляемых объектов - исходные данные и результаты выполнения задания.

`System.Guid SessionGuid`

GUID сессии, в рамках которой выполняется работа.

`System.Threading.Thread Thread`

Поток, в котором выполняется задание.

7.5.14.3 Свойства класса

`Intermech.Interfaces.DeleteAnalyzerOptions Options { get; }`

Параметры.

`Intermech.Interfaces.DeleteAnalyzerJobStatus Status { get; }`

Состояние задания по анализу списка удаляемых объектов (точная копия состояния).

7.5.14.4 Методы класса

DeleteAnalyzerJob (конструктор)

Создать задание по анализу списка удаляемых объектов.

Синтаксис:

```
void DeleteAnalyzerJob (System.Guid sessionGuid; System.Collections.Generic.Dictionary<Guid,
IObjectsDeleteAnalyzer> analyzers;
```

```
Intermech.Interfaces.DeletingObjects items; Intermech.Interfaces.DeleteAnalyzerOptions options)
```

Параметры:

`sessionGuid`

Идентификатор сессии, в рамках которой выполняется задание.

`analyzers`

Коллекция анализаторов, к которым будет обращаться задание в ходе выполнения анализа.

`items`

Список описаний удаляемых объектов - исходные данные и результаты выполнения задания.

`options`

Параметры.

Equals

Выполнить сравнение с указанным объектом.

Синтаксис:

```
bool Equals (object obj)
```

Параметры:

`obj`

Объект для сравнения.

Возвращаемое значение:

true, если объекты равны.

GetHashCode

Получить 32-битный хэш-код экземпляра объекта.

Синтаксис:

```
Int32 GetHashCode ()
```

Возвращаемое значение:

32-битный хэш-код экземпляра объекта.

ThreadMethod

Основной метод потока.

Синтаксис:

```
protected virtual void ThreadMethod ()
```

7.5.15 Класс ObjectsDeleteAnalyzerService

Серверная служба анализаторов списка удаляемых объектов.

7.5.15.1 Пространство имен

Intermech.Kernel

7.5.15.2 Реализуемые интерфейсы:

Intermech.Interfaces.IObjectsDeleteAnalyzerService

7.5.15.3 Поля класса:

`System.Collections.Generic.Dictionary<Guid, IObjectsDeleteAnalyzer> analyzers`

Коллекция анализаторов в коллекции.

`System.Collections.Generic.Dictionary<Guid, DeleteAnalyzerJob> jobs`

Коллекция выполняющихся заданий.

7.5.15.4 Методы класса:

ObjectsDeleteAnalyzerService (конструктор)

Создать экземпляр серверной службы анализаторов.

Синтаксис:

`void ObjectsDeleteAnalyzerService ()`

Analyze

Начать выполнение анализа удаляемых объектов, при необходимости добавить в граф дополнительные идентификаторы версий объектов, которые тоже требуется удалить. На верхнем уровне - первоначальный список удаляемых версий объектов. Служба опрашивает все зарегистрированные в ней анализаторы для выполнения данного анализа. Вся работа выполняется на сервере в отдельном потоке, в рамках задания по анализу.

Синтаксис:

`System.Guid Analyze (System.Guid sessionGuid; Intermech.Interfaces.DeletingObjects deletingObjects; Intermech.Interfaces.DeleteAnalyzerOptions options)`

Параметры:

`sessionGuid`

идентификатор сессии, в рамках которой выполняется анализ.

`deletingObjects`

Список удаляемых версий объектов.

`options`

Параметры.

Возвращаемое значение:

Уникальный идентификатор задания, в рамках которого выполняется анализ. Значение `Guid.Empty` означает невозможность начать анализ.

CancelJob

Прервать указанное задание.

Синтаксис:

`bool CancelJob (System.Guid jobID)`

Параметры:

`jobID`

Задание.

Возвращаемое значение:

`true`, если задание было найдено и остановлено.

LoadDescriptions

Заполнить описания в указанных объектах.

Синтаксис:

`Intermech.Interfaces.DeletingObjects LoadDescriptions (System.Guid sessionGuid; Intermech.Interfaces.DeletingObjects deletingObjects)`

Параметры:

`sessionGuid`

идентификатор сессии, в рамках которой выполняется анализ.

`deletingObjects`

Список удаляемых версий объектов.

Возвращаемое значение:

Список удаляемых объектов, у которых заполнены все поля.

QueryJobStatus

Запросить статус указанного задания на сервере. Если задание успешно или ошибочно завершено, вернётся полный пакет со статусом задания, а само задание будет удалено на серверной стороне вместе со своим потоком.

Синтаксис:

`Intermech.Interfaces.DeleteAnalyzerJobStatus QueryJobStatus (System.Guid jobID)`

Параметры:

`jobID`

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

RegisterAnalyzer

Выполнить регистрацию анализатора в службе.

Синтаксис:

`bool RegisterAnalyzer (Intermech.Interfaces.IObjectsDeleteAnalyzer analyzer)`

Параметры:

`analyzer`

Анализатор.

Возвращаемое значение:

`true`, если регистрация выполнена успешно.

UnregisterAnalyzer

Выполнить удаление анализатора из службы.

Синтаксис:

`bool UnregisterAnalyzer (Intermech.Interfaces.IObjectsDeleteAnalyzer analyzer)`

Параметры:

`analyzer`

Анализатор.

Возвращаемое значение:

`true`, если удаление выполнено успешно.

UnregisterAnalyzer

Выполнить удаление анализатора по его Guid из службы.

Синтаксис:

`bool UnregisterAnalyzer (System.Guid analyzerGuid)`

Параметры:

`analyzerGuid`

Guid анализатора.

Возвращаемое значение:

`true`, если удаление выполнено успешно.

7.6 Удаление объектов

Серверная часть IPS Development Kit предоставляет разработчикам возможность пакетного удаления списка объектов и связей. Для этих целей используется серверная служба `IObjectsDeleteService`, которая будет подробно описана в следующем подразделе.

Перед выполнением удаления требуется сформировать список удаляемых объектов и связей. Для этих целей используется служба `IObjectsDeleteAnalyzerService`, которая подробно описана в разделе [7.5.1](#).

7.6.1 Интерфейс `IObjectsDeleteService`

Интерфейс серверной службы по удалению объектов.

7.6.1.1 Пространство имен

Intermech.Interfaces

7.6.1.2 Методы интерфейса:

CancelJob

Прервать указанное задание.

Синтаксис:

`Intermech.Interfaces.DeleteObjectsJobStatus CancelJob (System.Guid jobID)`

Параметры:

`jobID`

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

Delete

Начать выполнение удаления объектов. Вся работа выполняется на сервере в отдельном потоке.

Синтаксис:

`System.Guid Delete (System.Guid sessionGuid; Intermech.Interfaces.DeletingObjects deletingObjects; Intermech.Interfaces.DeleteObjectsJobMode mode)`

Параметры:

`sessionGuid`

идентификатор сессии, в рамках которой выполняется удаление.

`deletingObjects`

Список удаляемых объектов.

`mode`

Режим удаления объектов.

Возвращаемое значение:

Уникальный идентификатор задания, в рамках которого выполняется удаление объектов. Значение `Guid.Empty` означает невозможность начать анализ.

PauseJob

Приостановить указанное задание.

Синтаксис:

`Intermech.Interfaces.DeleteObjectsJobStatus PauseJob (System.Guid jobID)`

Параметры:

`jobID`

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

QueryJobStatus

Запросить статус указанного задания на сервере. Если задание успешно или ошибочно завершено, вернётся полный пакет со статусом задания, а само задание будет удалено на серверной стороне вместе со своим потоком.

Синтаксис:

`Intermech.Interfaces.DeleteObjectsJobStatus QueryJobStatus (System.Guid jobID)`

Параметры:

`jobID`

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

ResumeJob

Возобновить указанное задание в указанном режиме.

Синтаксис:

`Intermech.Interfaces.DeleteObjectsJobStatus ResumeJob (System.Guid jobID; Intermech.Interfaces.DeleteObjectsJobMode mode)`

Параметры:

jobID

Задание.

mode

Режим удаления объектов.

Возвращаемое значение:

Статус указанного задания на сервере.

7.6.2 Перечислитель DeleteObjectsJobMode

Режим удаления объектов.

7.6.2.1 Пространство имен

Intermech.Interfaces

7.6.2.2 Значения перечислителя:

AbortOnError = 0x00000001

Прерывать удаление при возникновении ошибки (полностью завершать задачу по удалению).

AscOnError = 0x00000000

Запросить действие при возникновении ошибки (по умолчанию).

IgnoreErrors = 0x00000002

Удалять всё, что можно, игнорируя ошибки.

7.6.3 Перечислитель DeleteObjectsJobProgress

Индикатор выполнения задания.

7.6.3.1 Пространство имен

Intermech.Interfaces

7.6.3.2 Значения перечислителя:

Cancelled = 0xFFFFFFFF

Задание прервано пользователем.

Completed = 0x00000002

Задание успешно завершено.

Error = 0xFFFFFFFFE

Задание прервано из-за ошибки.

Idle = 0xFFFFFFFDD

Задание простаивает, ожидая действие со стороны клиента.

NotStarted = 0x00000000

Задание ещё не запущено.

Working = 0x00000001

Задание работает.

7.6.4 Класс DeleteObjectsJobStatus

Состояние задания по удалению объектов.

7.6.4.1 Пространство имен

Intermech.Interfaces

7.6.4.2 Реализуемые интерфейсы:

System.ICloneable

7.6.4.3 Поля класса:

System.Exception Exception

Исключение.

System.Collections.Generic.List<Int64> Items

Список идентификаторов версий удалённых объектов. Значение будет заполнено только когда задание будет успешно выполнено.

Intermech.Interfaces.DeleteObjectsJobMode Mode
 Режим удаления объектов.
 Int32 Objects
 Количество объектов, успешно удалённых.
 Intermech.Interfaces.DeleteObjectsJobProgress Progress
 Индикатор выполнения задания.
 System.Collections.Generic.List<Int64> Relations
 Список идентификаторов удалённых связей. Значение будет заполнено только когда задание будет успешно выполнено.
 Int32 RelationsCount
 Количество связей, успешно удалённых.
 System.Collections.Generic.List<Int64> RelationsProjIDs
 Список идентификаторов родительских объектов удалённых связей. Значение будет заполнено только когда задание будет успешно выполнено.
 System.Collections.Generic.List<Int32> RelationsTypeIDs
 Список идентификаторов типов удалённых связей. Значение будет заполнено только когда задание будет успешно выполнено.
 Int32 Skipped
 Количество объектов, удаление которых не было выполнено.

7.6.4.4 Методы класса:

DeleteObjectsJobStatus (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void DeleteObjectsJobStatus ()
```

Cancel

Установить поля класса в "Задание прервано".

Синтаксис:

```
void Cancel (Int32 objects; Int32 skipped; Int32 relationsCount;
System.Collections.Generic.List<Int64> items;
System.Collections.Generic.List<Int64> relations;
System.Collections.Generic.List<Int64> relProjIDs;
System.Collections.Generic.List<Int64> relTypeIDs)
```

Параметры:

objects

Количество добавленных описаний объектов.

skipped

Количество объектов, удаление которых не было выполнено.

relationsCount

Количество удалённых связей.

items

Коллекция идентификаторов успешно удалённых объектов.

relations

Список идентификаторов удалённых связей.

relProjIDs

Список идентификаторов версий родительских объектов удалённых связей.

relTypeIDs

Список типов удалённых связей.

Clone

Создать копию экземпляра класса.

Синтаксис:

```
object Clone ()
```

Возвращаемое значение:

Копия экземпляра класса.

Complete

Установить поля класса в "Задание успешно выполнено".

Синтаксис:

```
void Complete (Int32 objects; Int32 skipped; Int32 relationsCount;  
System.Collections.Generic.List<Int64> items;  
System.Collections.Generic.List<Int64> relations;  
System.Collections.Generic.List<Int64> relProjIDs;  
System.Collections.Generic.List<Int64> relTypeIDs)
```

Параметры:

objects

Количество добавленных описаний объектов.

skipped

Количество объектов, удаление которых не было выполнено.

relationsCount

Количество удалённых связей.

items

Коллекция идентификаторов успешно удалённых объектов.

relations

Список идентификаторов удалённых связей.

relProjIDs

Список идентификаторов версий родительских объектов удалённых связей.

relTypeIDs

Список типов удалённых связей.

Error

Установить поля класса в "Задание остановлено из-за ошибки".

Синтаксис:

```
void Error (System.Exception exception; Int32 objects;  
Int32 skipped; Int32 relationsCount;  
System.Collections.Generic.List<Int64> items;  
System.Collections.Generic.List<Int64> relations;  
System.Collections.Generic.List<Int64> relProjIDs;  
System.Collections.Generic.List<Int64> relTypeIDs)
```

Параметры:

exception

Возникшее исключение.

objects

Количество добавленных описаний объектов.

skipped

Количество объектов, удаление которых не было выполнено.

relationsCount

Количество удалённых связей.

items

Коллекция идентификаторов успешно удалённых объектов.

relations

Список идентификаторов удалённых связей.

relProjIDs

Список идентификаторов версий родительских объектов удалённых связей.

relTypeIDs

Список типов удалённых связей.

Pause

Приостановить задание, ожидая реакцию со стороны клиента.

Синтаксис:

```
void Pause (System.Exception exception;  
Int32 objects; Int32 skipped; Int32 relationsCount;
```

```
System.Collections.Generic.List<Int64> items;
System.Collections.Generic.List<Int64> relations;
System.Collections.Generic.List<Int64> relProjIDs;
System.Collections.Generic.List<Int64> relTypeIDs;
Intermech.Interfaces.DeleteObjectsJobMode mode)
```

Параметры:

exception

Возникшее исключение.

objects

Количество добавленных описаний объектов.

skipped

Количество объектов, удаление которых не было выполнено.

relationsCount

Количество удалённых связей.

items

Коллекция идентификаторов успешно удалённых объектов.

relations

Список идентификаторов удалённых связей.

relProjIDs

Список идентификаторов версий родительских объектов удалённых связей.

relTypeIDs

Список типов удалённых связей.

mode

Режим удаления объектов.

Start

Установить поля класса в "Задание стартовало".

Синтаксис:

```
void Start ()
```

7.6.5 Класс DeleteObjectsJob

Задание по пакетному удалению объектов, выполняемое на сервере в отдельном потоке.

7.6.5.1 Пространство имен

Intermech.Kernel

7.6.5.2 Поля класса:

```
System.Collections.Generic.List<Int64> DeletedItems
```

Список идентификаторов версий удаляемых объектов - список успешно удалённых объектов.

```
System.Collections.Generic.List<Int64> DeletedRelations
```

Список идентификаторов удаляемых связей - список успешно удалённых связей.

```
System.Collections.Generic.List<Int64> DeletedRelationsProjIDs
```

Список идентификаторов версий родительских объектов удаляемых связей.

```
System.Collections.Generic.List<Int32> DeletedRelationsTypeIDs
```

Список идентификаторов типов удаляемых связей.

```
System.Guid Guid
```

Уникальный идентификатор задания.

```
Intermech.Interfaces.DeletingObjects Items
```

Список удаляемых объектов - исходные данные.

```
System.Guid SessionGuid
```

GUID сессии, в рамках которой выполняется работа.

```
System.Threading.Thread Thread
```

Поток, в котором выполняется задание.

7.6.5.3 Свойства класса:

```
Intermech.Interfaces.DeleteObjectsJobStatus Status { get; }
```

Состояние задания по пакетному удалению объектов (точная копия состояния).

7.6.5.4 Методы класса:

DeleteObjectsJob (конструктор)

Создать задание по анализу списка удаляемых объектов.

Синтаксис:

```
void DeleteObjectsJob (System.Guid sessionGuid; Intermech.Interfaces.DeletingObjects items)
```

Параметры:

sessionGuid

Идентификатор сессии, в рамках которой выполняется задание.

items

Список идентификаторов версий удаляемых объектов - исходные данные и результаты выполнения задания.

Equals

Выполнить сравнение с указанным объектом.

Синтаксис:

```
bool Equals (object obj)
```

Параметры:

obj

Объект для сравнения.

Возвращаемое значение:

true, если объекты равны.

GetHashCode

Получить 32-битный хэш-код экземпляра объекта.

Синтаксис:

```
Int32 GetHashCode ()
```

Возвращаемое значение:

32-битный хэш-код экземпляра объекта.

SleepIfNeed

Поток находится в режиме ожидания до тех пор, пока ему не изменят статус.

Синтаксис:

```
protected virtual void SleepIfNeed ()
```

ThreadMethod

Основной метод потока.

Синтаксис:

```
protected virtual void ThreadMethod ()
```

7.6.6 Класс ObjectsDeleteService

Серверная служба для пакетного удаления объектов.

7.6.6.1 Пространство имен

Intermech.Kernel

7.6.6.2 Реализуемые интерфейсы:

Intermech.Interfaces.IObjectsDeleteService

7.6.6.3 Поля класса:

```
System.Collections.Generic.Dictionary<Guid, DeleteObjectsJob> jobs
```

Коллекция выполняющихся заданий.

7.6.6.4 Методы класса:

ObjectsDeleteService (конструктор)

Создать экземпляр серверной службы анализаторов.

Синтаксис:

```
void ObjectsDeleteService ()
```

CancelJob

Прервать указанное задание.

Синтаксис:

```
Intermech.Interfaces.DeleteObjectsJobStatus CancelJob (System.Guid jobID)
```

Параметры:

jobID

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

Delete

Начать выполнение удаления объектов. Вся работа выполняется на сервере в отдельном потоке.

Синтаксис:

```
System.Guid Delete (System.Guid sessionGuid;  
Intermech.Interfaces.DeletingObjects deletingObjects;  
Intermech.Interfaces.DeleteObjectsJobMode mode)
```

Параметры:

sessionGuid

идентификатор сессии, в рамках которой выполняется удаление.

deletingObjects

Список удаляемых версий объектов.

mode

Режим удаления объектов.

Возвращаемое значение:

Уникальный идентификатор задания, в рамках которого выполняется удаление объектов. Значение Guid.Empty означает невозможность начать анализ.

PauseJob

Приостановить указанное задание.

Синтаксис:

```
Intermech.Interfaces.DeleteObjectsJobStatus PauseJob (System.Guid jobID)
```

Параметры:

jobID

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

QueryJobStatus

Запросить статус указанного задания на сервере. Если задание успешно или ошибочно завершено, вернется полный пакет со статусом задания, а само задание будет удалено на серверной стороне вместе со своим потоком.

Синтаксис:

```
Intermech.Interfaces.DeleteObjectsJobStatus QueryJobStatus (System.Guid jobID)
```

Параметры:

jobID

Задание.

Возвращаемое значение:

Статус указанного задания на сервере.

ResumeJob

Возобновить указанное задание в указанном режиме.

Синтаксис:

```
Intermech.Interfaces.DeleteObjectsJobStatus ResumeJob (System.Guid jobID;  
Intermech.Interfaces.DeleteObjectsJobMode mode)
```

Параметры:

jobID

Задание.

mode

Режим удаления объектов.
Возвращаемое значение:
Статус указанного задания на сервере.

7.7 Пакетное взятие объектов на изменение

Ядро IPS Development Kit предоставляет возможность взятия на изменение группы информационных объектов. Для этих целей используется серверная служба IObjectsCheckOutServerService. Далее будут подробно описаны служба и все вспомогательные классы, которые она использует.

7.7.1 Интерфейс IObjectsCheckOutServerService

Серверный сервис, позволяющий брать на изменение группы объектов.

7.7.1.1 Пространство имен

Intermech.Interfaces

7.7.1.2 Методы интерфейса:

CheckOut

Получить список версий объектов для редактирования.

Синтаксис:

```
System.Collections.Generic.List<ObjectCheckOutVersionDescription> CheckOut (object usrSession;  
System.Collections.Generic.IList<Int64> versions;  
bool throwException)
```

Параметры:

usrSession

Пользовательская сессия. При обращении к сервису со стороны сервера сюда можно передавать ссылку на интерфейс IUserSession или GUID сессии (как строку или System.Guid). При обращении к сервису со стороны клиента сюда можно передавать только GUID сессии (как строку или System.Guid).

versions

Список версий объектов, которые требуется взять на изменение.

throwException

true - при ошибке сгенерировать исключение.

Возвращаемое значение:

Список описаний версий объектов для редактирования. Если возникла ошибка, будет возвращено значение null.

LoadDescriptions

Загрузить описания указанных версий объектов.

Синтаксис:

```
System.Collections.Generic.List<ObjectCheckOutVersionDescription> LoadDescriptions (object usrSession;  
System.Collections.Generic.IList<Int64> versions;  
bool throwException)
```

Параметры:

usrSession

Пользовательская сессия. При обращении к сервису со стороны сервера сюда можно передавать ссылку на интерфейс IUserSession или GUID сессии (как строку или System.Guid). При обращении к сервису со стороны клиента сюда можно передавать только GUID сессии (как строку или System.Guid).

versions

Список идентификаторов версий объектов.

throwException

true - при ошибке сгенерировать исключение.

Возвращаемое значение:

Список описаний версий объектов. Если возникла ошибка, будет возвращено значение null.

Rollback

Выполнить откат произведённых изменений по списку-результату метода CheckOut.

Синтаксис:

```
void Rollback (object usrSession;
```

```
System.Collections.Generic.List<ObjectCheckOutVersionDescription> rollback;  
bool throwException)
```

Параметры:

usrSession

Пользовательская сессия. При обращении к сервису со стороны сервера сюда можно передавать ссылку на интерфейс IUserSession или GUID сессии (как строку или System.Guid). При обращении к сервису со стороны клиента сюда можно передавать только GUID сессии (как строку или System.Guid).

rollback

Список редактируемых версий объектов, полученных в методе CheckOut.

throwException

true - при ошибке сгенерировать исключение.

7.7.2 Интерфейс IAssignable

Интерфейс позволяет копировать в поля объекта содержимое полей другого объекта.

7.7.2.1 Пространство имен

Intermech.Interfaces

7.7.2.2 Методы интерфейса:

Assign

Скопировать в текущий объект поля из другого объекта.

Синтаксис:

```
void Assign (object source)
```

Параметры:

source

Объект-источник.

Clear

Очистить поля класса.

Синтаксис:

```
void Clear ()
```

7.7.3 Класс ObjectVersionDescription

Класс-контейнер, в котором хранится описание версии объекта.

7.7.3.1 Пространство имен

Intermech.Interfaces

7.7.3.2 Реализуемые интерфейсы:

Intermech.Interfaces.IAssignable

System.Collections.Generic.IComparer<ObjectVersionDescription>

System.ICloneable

System.IComparable

System.IComparable<ObjectVersionDescription>

7.7.3.3 Статические поля класса:

```
static System.Collections.Generic.List<ColumnDescriptor> columnDescriptors
```

Список колонок для запроса в ядро.

7.7.3.4 Поля класса:

```
string CAPTION
```

Заголовок.

```
Int64 F_BASE_VERSION
```

Признак базовой версии.

```
Int64 F_CHKOUT_BY
```

Кем взят на изменение.

```
Int64 F_ID
```

Идентификатор объекта.

Int32 F_LCLEVEL_ID
Идентификатор уровня продвижения для шага ЖЦ версии объекта.

Int32 F_LCSCHEMA_ID
Идентификатор схемы ЖЦ для шага ЖЦ версии объекта.

Int32 F_LCSTEP_ID
Шаг ЖЦ.

Int64 F_MODIFICATION_ID
Номер группы изменений.

Int64 F_OBJECT_ID
Идентификатор версии объекта.

Int32 F_OBJECT_TYPE
Идентификатор типа объекта.

Int64 F_OWNER_ID
Владелец объекта.

Int64 F_VERSION_ID
Номер версии.

Intermech.Interfaces.ObjectVersionDescriptionOptions Options
Дополнительная информация о версии объекта из контекста редактирования.

7.7.3.5 Методы класса:

ObjectVersionDescription (конструктор)

Создать экземпляр класса, заполнить его информацией из указанного объекта.

Синтаксис:

```
void ObjectVersionDescription (Intermech.Interfaces.IDBObject source)
```

Параметры:

source

Объект-описатель.

ObjectVersionDescription (конструктор)

Создать экземпляр класса, заполнить его информацией из строки таблицы.

Синтаксис:

```
void ObjectVersionDescription (System.Data.DataRow row)
```

Параметры:

row

Строка таблицы с данными.

ObjectVersionDescription (конструктор)

Создать заполненный экземпляр класса.

Синтаксис:

```
void ObjectVersionDescription (Int64 _ID; Int64 _OBJECT_ID; Int32 _OBJECT_TYPE;  
Int32 _LCSTEP_ID; Int64 _OWNER_ID; Int64 _CHKOUT_BY; string _CAPTION;  
Int64 _F_VERSION_ID; Int64 _F_MODIFICATION_ID; Int64 _F_BASE_VERSION;  
Intermech.Interfaces.ObjectVersionDescriptionOptions _Options)
```

Параметры:

_ID

Идентификатор объекта.

_OBJECT_ID

Идентификатор версии объекта.

_OBJECT_TYPE

Идентификатор типа объекта.

_LCSTEP_ID

Шаг ЖЦ.

_OWNER_ID

Владелец объекта.

_CHKOUT_BY

Владелец объекта.

_CAPTION

Заголовок.

_F_VERSION_ID

Номер версии.

_F_MODIFICATION_ID

Номер группы изменений.

_F_BASE_VERSION

Признак базовой версии.

_Options

Опции.

ObjectVersionDescription (конструктор)

Создать экземпляр класса, заполнить его информацией из указанного объекта.

Синтаксис:

```
void ObjectVersionDescription (object source)
```

Параметры:

source

Источник информации.

Assign

Скопировать информацию из указанного объекта.

Синтаксис:

```
void Assign (object source)
```

Параметры:

source

Объект-источник.

CalcFields

Определить значения рассчитываемых полей.

Синтаксис:

```
void CalcFields ()
```

Clear

Очистить экземпляр класса.

Синтаксис:

```
void Clear ()
```

Clone

Вернуть точную копию экземпляра класса.

Синтаксис:

```
object Clone ()
```

Возвращаемое значение:

Точная копия экземпляра класса.

Compare

Сравнить два объекта.

Синтаксис:

```
Int32 Compare (Intermech.Interfaces.ObjectVersionDescription x;  
Intermech.Interfaces.ObjectVersionDescription y)
```

Параметры:

x

Первый объект.

y

Второй объект.

Возвращаемое значение:

-1, 0, 1.

Compare

Сравнить два объекта.

Синтаксис:

```
Int32 Compare (object x; object y)
```

Параметры:

x

Первый объект.

y

Второй объект.

Возвращаемое значение:

-1, 0, 1.

CompareTo

Сравнить с указанным объектом.

Синтаксис:

```
Int32 CompareTo (Intermech.Interfaces.ObjectVersionDescription other)
```

Параметры:

other

Объект для сравнения.

Возвращаемое значение:

-1, 0, 1.

CompareTo

Сравнить с указанным объектом.

Синтаксис:

```
Int32 CompareTo (object obj)
```

Параметры:

obj

Объект для сравнения.

Возвращаемое значение:

-1, 0, 1.

Equals

Сравнить с указанным объектом.

Синтаксис:

```
bool Equals (object obj)
```

Параметры:

obj

Объект для сравнения.

Возвращаемое значение:

true, если объекты равны.

GetColumnDescriptors

Получить список колонок, необходимых для получения списка объектов.

Синтаксис:

```
System.Collections.Generic.List<ColumnDescriptor> GetColumnDescriptors ()
```

Возвращаемое значение:

Список колонок, необходимых для получения списка объектов.

GetHashCode

Получить 32-битный хэш-код экземпляра класса.

Синтаксис:

```
Int32 GetHashCode ()
```

Возвращаемое значение:

32-битный хэш-код экземпляра класса.

ToString

Получить представление экземпляра класса в виде строки.

Синтаксис:

```
string ToString ()
```

Возвращаемое значение:

Строковое представление экземпляра класса.

7.7.4 Класс ObjectCheckOutVersionDescription

Класс-контейнер, в котором хранится описание версии объекта для взятия на изменение.

7.7.4.1 Пространство имен

Intermech.Interfaces

7.7.4.2 Реализуемые интерфейсы:

Intermech.Interfaces.IAssignable
System.Collections.Generic.IComparer<ObjectCheckOutVersionDescription>
System.ICloneable
System.IComparable
System.IComparable<ObjectCheckOutVersionDescription>

7.7.4.3 Поля класса:

Intermech.Interfaces.ObjectCheckedOutVersionMode Mode

Каким образом была получена версия для редактирования.

7.7.4.4 Методы класса:

ObjectCheckOutVersionDescription (конструктор)

Создать экземпляр класса, заполнить его информацией из указанного объекта.

Синтаксис:

```
void ObjectCheckOutVersionDescription (Intermech.Interfaces.IDBObject source)
```

Параметры:

source

Объект-описатель.

ObjectCheckOutVersionDescription (конструктор)

Создать экземпляр класса, заполнить его информацией из строки таблицы.

Синтаксис:

```
void ObjectCheckOutVersionDescription (System.Data.DataRow row)
```

Параметры:

row

Строка таблицы с данными.

ObjectCheckOutVersionDescription (конструктор)

Создать заполненный экземпляр класса.

Синтаксис:

```
void ObjectCheckOutVersionDescription (Int64 _ID; Int64 _OBJECT_ID;  
Int32 _OBJECT_TYPE; Int32 _LCSTEP_ID;  
Int64 _OWNER_ID; Int64 _CHKOUT_BY; string _CAPTION; Int64 _F_VERSION_ID;  
Int64 _F_MODIFICATION_ID; Int64 _F_BASE_VERSION;  
Intermech.Interfaces.ObjectVersionDescriptionOptions _Options;  
Intermech.Interfaces.ObjectCheckedOutVersionMode _Mode)
```

Параметры:

_ID

Идентификатор объекта.

_OBJECT_ID

Идентификатор версии объекта.

_OBJECT_TYPE

Идентификатор типа объекта.

_LCSTEP_ID

Шаг ЖЦ.

_OWNER_ID

Владелец объекта.

_CHKOUT_BY

Владелец объекта.

`_CAPTION`

Заголовок.

`_F_VERSION_ID`

Номер версии.

`_F_MODIFICATION_ID`

Номер группы изменений.

`_F_BASE_VERSION`

Признак базовой версии.

`_Options`

Опции.

`_Mode`

Каким образом была получена версия для редактирования.

ObjectCheckOutVersionDescription (конструктор)

Создать экземпляр класса, заполнить его информацией из указанного объекта.

Синтаксис:

```
void ObjectCheckOutVersionDescription (object source)
```

Параметры:

`source`

Источник информации.

Assign

Скопировать информацию из указанного объекта.

Синтаксис:

```
void Assign (object source)
```

Параметры:

`source`

Объект-источник.

Clear

Очистить экземпляр класса.

Синтаксис:

```
void Clear ()
```

7.7.5 Перечислитель ObjectVersionDescriptionOptions

Дополнительная информация о версии объекта из контекста редактирования.

7.7.5.1 Пространство имен

Intermech.Interfaces

7.7.5.2 Значения перечислителя:

`FromECOCComposition = 0x10`

Данное описание принадлежит версии объекта, которая входит в состав извещения.

`InvalidDescription = 0x2000000000000000`

Описание неполное или некорректное.

`IsContext = 0x1`

Данное описание принадлежит непосредственно контексту редактирования, а не объекту из контекста.

`IsECO = 0x2`

Данное описание принадлежит непосредственно контексту редактирования, а не объекту из контекста, причём контекст является извещением.

`None = 0x0`

Нет никакой дополнительной информации о версии объекта из контекста редактирования.

7.7.6 Перечислитель ObjectCheckedOutVersionMode

Каким образом была получена версия для редактирования.

7.7.6.1 Пространство имен

Intermech.Interfaces

7.7.6.2 Значения перечислителя:

ActualCopy = 0x00000002

Существующая рабочая копия указанной версии объекта.

CheckOut = 0x00000003

Версия объекта была взята на изменение.

Error = 0xFFFFFFFF

Возникла ошибка при получении версии для редактирования.

InBase = 0x00000001

Объект можно модифицировать в базе данных.

NewVersion = 0x00000004

Выпущена новая версия для редактирования.

None = 0x00000000

Нет информации или исходная версия.

7.7.7 Класс ObjectVersionDescriptionsHelper

Статический вспомогательный класс, позволяющий загружать описания для версий объектов. Класс позволяет загружать описания в списки любых типов, унаследованных от типа ObjectVersionDescription.

7.7.7.1 Пространство имен

Intermech.Interfaces

7.7.7.2 Статические методы класса

LoadDescription

Загрузить описания указанной версии объекта.

Синтаксис:

```
static object LoadDescription (Intermech.Interfaces.IUserSession session;  
System.Type T;  
Int64 version)
```

Параметры:

session

Сессия.

T

Тип, унаследованный от ObjectVersionDescription.

version

Идентификатор версии объекта, для которой надо загрузить описание.

Возвращаемое значение:

Описание указанной версии объекта.

LoadDescriptions

Загрузить описания указанных версий объектов.

Синтаксис:

```
static System.Collections.Generic.List<object> LoadDescriptions (Intermech.Interfaces.IUserSession  
session;
```

System.Type T;

System.Collections.Generic.IList<Int64> versions;

Int32 objectType)

Параметры:

session

Сессия.

T

Тип, унаследованный от ObjectVersionDescription.

versions

Список версий, для которых надо загрузить описания.

objectType

Если все версии принадлежат одному типу объекта, или имеют общий родительский тип, его обязательно надо указать, чтобы ядро могло использовать свою оптимизацию. Если тип неизвестен, следует указать -1.

Возвращаемое значение:

Отсортированный список описаний версий объектов.

LoadDescriptions

Загрузить описания версий указанного объекта.

Синтаксис:

```
static System.Collections.Generic.List<object> LoadDescriptions (Intermech.Interfaces.IUserSession session;
```

```
System.Type T;
```

```
Int64 F_ID;
```

```
Int32 objectType)
```

Параметры:

session

Сессия.

T

Тип, унаследованный от ObjectVersionDescription.

F_ID

Идентификатор объекта, для которого требуется получить описания его версий.

objectType

Если все версии принадлежат одному типу объекта, или имеют общий родительский тип, его обязательно надо указать, чтобы ядро могло использовать свою оптимизацию. Если тип неизвестен, следует указать -1.

Возвращаемое значение:

Отсортированный список описаний версий объектов.

LoadUnsortedDescriptions

Загрузить описания указанных версий объектов.

Синтаксис:

```
static System.Collections.Generic.List<object> LoadUnsortedDescriptions (Intermech.Interfaces.IUserSession session;
```

```
System.Type T;
```

```
System.Collections.Generic.IList<Int64> versions;
```

```
Int32 objectType)
```

Параметры:

session

Сессия.

T

Тип, унаследованный от ObjectVersionDescription.

versions

Список версий, для которых надо загрузить описания.

objectType

Если все версии принадлежат одному типу объекта, или имеют общий родительский тип, его обязательно надо указать, чтобы ядро могло использовать свою оптимизацию. Если тип неизвестен, следует указать -1.

Возвращаемое значение:

Неотсортированный список описаний версий объектов.

7.8 Индексация содержимого документов

IPS Development Kit ведёт общий поисковый индекс, в который попадают значения атрибутов, у которых установлен флаг `AttributeOptions.AddToGlobalIndex`. Поиск по общему индексу ведётся с помощью оператора `RelationalOperators.InGlobalIndex`, а искомое значение задаётся в условии поиска с помощью класса `GlobalIndexSearchValue`, описанного в пространстве имён `Intermech.Kernel.Search`. Индекс ведётся только по атрибутам объектов. Индексировать можно атрибуты строкового, текстового и файлового типа. В последнем случае индексируется как имя файла, так и его содержимое. С этой целью на сервере приложений можно регистрировать файловые конвертеры, предназначенные для извлечения из файлов различных форматов текстовой информации, предназначенной для помещения в поисковый индекс. По умолчанию с системой

поставляются конвертеры для текстовых файлов, документов встроенного редактора (ведомости, извещения, спецификации, отчёты и пр.), файлов DWG, DXF и некоторых других форматов. В системе можно также регистрировать конвертеры для индексации файлов собственных форматов. Индексируются только те файлы, для которых система сможет найти соответствующий конвертер. Ниже описан процесс создания и подключения собственного конвертера файлов.

7.8.1 Написание конвертера файлов

Конвертер файлов должен быть реализован в серверном плагине и должен поддерживать интерфейс `IndexerFileConverter`, описанный в пространстве имён `Intermech.Interfaces.Server.GlobalIndex`. Часть методов данного интерфейса реализована в классе `Intermech.Kernel.GlobalIndex.CustomFileConverter`, поэтому рекомендуется наследовать свой конвертер от данного класса.

7.8.1.1 Интерфейс `IndexerFileConverter`

Данный интерфейс должен поддерживать конвертер файлов, т.к. методы этого интерфейса используются системой для регистрации конвертера и извлечения текста из файлов, поддерживаемых данным конвертером.

7.8.1.2 Пространство имен

`Intermech.Interfaces.Server.GlobalIndex`

7.8.1.3 Свойства интерфейса

`string Caption`

Наименование конвертера для отображения в различных списках и сообщениях. Свойство доступно только для чтения.

`int Priority`

Приоритет конвертера. Позволяет реализовывать конвертеры, перекрывающие уже зарегистрированные в системе. При поиске конвертера для файла система сначала опрашивает конвертеры с большим приоритетом. Системные конвертеры имеют приоритет 0. Свойство доступно только для чтения.

7.8.1.4 Методы интерфейса

`CanGetPlainText`

Метод определяет, может ли данный конвертер извлечь текст из файла, который содержится в текущем значении указанного атрибута.

Синтаксис:

```
bool CanGetPlainText (IDBAttribute attribute);
```

Параметры:

`attribute`

Файловый атрибут, текущее значение которого нужно проиндексировать.

Возвращаемое значение:

Возвращает `true`, если данный файл может быть проиндексирован данным конвертером. Если конвертер возвращает `false`, то система продолжит опрос остальных конвертеров для определения возможности индексации файла данного атрибута.

`GetPlainText`

Метод извлекает текст из файла, содержащегося в текущем значении атрибута.

Синтаксис:

```
string GetPlainText (IDBAttribute attribute);
```

Параметры:

`attribute`

Файловый атрибут, текущее значение которого нужно проиндексировать.

Возвращаемое значение:

Возвращает текст без форматирования в кодировке Unicode.

7.8.1.5 Класс `CustomFileConverter`

Базовый абстрактный класс, реализующий некоторые методы

7.8.1.6 Пространство имен

`Intermech.Kernel.GlobalIndex`

7.8.1.7 Свойства класса

`public abstract string Caption`

Наименование конвертера для отображения в различных списках и сообщениях.

`public virtual int Priority`

Приоритет конвертера. Позволяет реализовывать конвертеры, перекрывающие уже зарегистрированные в системе. При поиске конвертера для файла система сначала опрашивает конвертеры с БОЛЬШИМ приоритетом. Базовая реализация возвращает 0.

`public virtual string[] SupportedFileExtensions`

Свойство возвращает поддерживаемые данным конвертером расширения файлов (в верхнем регистре и с точкой перед расширением). Если конвертер определяет поддерживаемые форматы файлов по их расширению, то ему достаточно перекрыть данное свойство и вернуть массив поддерживаемых им расширений. Метод `CanGetPlainText` в этом случае перекрывать не нужно.

7.8.1.8 Методы класса

CanGetPlainText

Метод определяет, может ли данный конвертер извлечь текст из файла, который содержится в текущем значении указанного атрибута.

Синтаксис:

`public virtual bool CanGetPlainText (IDBAttribute attribute);`

Параметры:

`attribute`

Файловый атрибут, текущее значение которого нужно проиндексировать.

Возвращаемое значение:

Возвращает `true`, если данный файл может быть проиндексирован данным конвертером. Базовая реализация определяет возможность извлечения текста по расширению файла в атрибуте. Данный метод следует перекрывать в случае, если расширения файла не достаточно для определения его формата.

GetPlainText

Метод извлекает текст из файла, содержащегося в текущем значении атрибута.

Синтаксис:

`public abstract string GetPlainText (IDBAttribute attribute);`

Параметры:

`attribute`

Файловый атрибут, текущее значение которого нужно проиндексировать.

Возвращаемое значение:

Возвращает текст без форматирования в кодировке Unicode. Разделители слов в тексте, которые использует индексатор IPS Development Kit, описаны в массиве `IndexerAttributeSettings.DefaultDelimiterChars` (пространство имен `Intermech.Kernel.Search`).

ReadFromStream

Метод извлекает текст из потока.

Синтаксис:

`protected string ReadFromStream (Stream strm);`

Параметры:

`strm`

Поток, из которого нужно прочитать текст.

Возвращаемое значение:

Возвращает строку, прочитанную из потока.

7.8.2 Регистрация конвертера файлов

Собственный конвертер файлов следует зарегистрировать в момент загрузки плагина с помощью метода `RegisterFileConverter` службы `IGlobalIndexService` (пространство имён `Intermech.Interfaces.Server.GlobalIndex`). Ниже приведён пример регистрации конвертера файлов.

```
using Intermech.Kernel.GlobalIndex;
using Intermech.Interfaces.Server.GlobalIndex;
// Описываем свой конвертер
class MyConverter: CustomFileConverter
```



```

{
// Реализация методов интерфейса IIndexerFileConverter
...
}
...
// Загрузка плагина
public void Load(IServiceProvider serviceProvider)
{
...
IGlobalIndexService ind_srvc = serviceProvider.GetService(typeof(IGlobalIndexService)) as
IGlobalIndexService;
MyConverter myConverter = new MyConverter();
ind_srvc.RegisterFileConverter(myConverter);
}

```

7.8.3 Создание пользовательских процедур экспертной системы

Пользовательская процедура экспертной системы может вызываться из скрипта генерации документа для реализации расчетов, которые неудобно либо невозможно сделать посредством самого скрипта. Как правило, пользовательская процедура работает с глобальными таблицами объектов и связей. Она должна соответствовать следующему заголовку:

```

internal void CheckTechQuantities(object obj, Int64[] context, DataTable dTable, Int32 objType, Int32
attrType, object Value)
{
ExpertServer.TaskInfo ti = (ExpertServer.TaskInfo)obj;
if ((ti.savedData == null) || (ti.savedLinks == null)) // Глобальные таблицы объектов и связей
return;
...
}

```

Для регистрации пользовательской процедуры надо включить в код загрузки модуля (например, в обработчик события LoadComplete) вызов метода сервера экспертной системы RegUserProc:

```

// Получить службу сервера экспертной системы
IExpertServer iExpServ = ServerServices.GetService(typeof(IExpertServer)) as IExpertServer;
// Регистрация пользовательских процедур
if (iExpServ != null)
{
iExpServ.RegUserProc("Проверить количества (изделия и материалы)", CheckTechQuantities);
}

```

8 Службы и компоненты универсального клиента

8.1 Создание объектов и их версий

8.1.1 IObjectCreatorService

Интерфейс IObjectCreatorService службы создания объектов позволяет вызвать мастер создания объектов с помощью функций:

1. CreateObjectDialog() – вызов мастера создания с последующим выбором типа создаваемого объекта
2. CreateObjectByTypeDialog – вызов мастера создания для указанного типа объектов (есть overload)
3. CreateObjectByTemplate – вызов мастера создания объекта по прототипу (есть overload)

Также имеется две функции для регистрации и разрегистрации мастера создания объектов определенного типа для переопределения имеющегося по умолчанию диалога (RegisterCreatorCustomService и UnregisterCreatorCustomService соответственно), которые принимают в качестве входных параметров идентификатор типа объекта и тип класса, реализующего интерфейс IObjectCreatorCustomService.

Для того чтобы получить интерфейс IObjectCreatorService нужно иметь ссылку на интерфейс IServiceProvider, у которого нужно выполнить метод GetService, передав в качестве параметра тип нужного интерфейса(IObjectCreatorService). Результат тоже приводится к нужному типу(IObjectCreatorService).

8.1.1.1 Методы интерфейса

CreateObjectByTemplateDialog

Создание нового объекта по прототипу.

Синтаксис:

```
Int64 CreateObjectByTemplateDialog(IDBObject aTemplateObject);
Int64 CreateObjectByTemplateDialog(Int64 aTemplateObjectID);
Int64 CreateObjectByTemplateDialog(Guid aObjectGuid);
Int64 CreateObjectByTemplateDialog(IDBObject aTemplateObject, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectByTemplateDialog(Int64 aTemplateObjectID, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectByTemplateDialog(Guid aObjectGuid, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectByTemplateDialog(IDBObject aTemplateObject, ObjectRelationLink[] aObjRelations,
DateTime aStartDate);
Int64 CreateObjectByTemplateDialog(Int64 aTemplateObjectID, ObjectRelationLink[] aObjRelations,
DateTime aStartDate);
Int64 CreateObjectByTemplateDialog(Guid aObjectGuid, ObjectRelationLink[] aObjRelations, DateTime
aStartDate);
```

Параметры:

aTemplateObject

Прототип объекта, по которому будет создан новый экземпляр.

aTemplateObjectID

Идентификатор, который задает объект-прототип для создаваемого экземпляра.

aObjectGuid

Глобальный идентификатор объекта, который является прототипом для создаваемого экземпляра.

aObjRelations

Массив структур, содержащих описание связей, которые нужно создать для нового объекта.

aStartDate

Дата, с которой начинают действовать созданные связи.

Возвращаемое значение:

Идентификатор созданного объекта. Возвращается -1, если объект не создан.

CreateObjectByTypeDialog

Создание нового объекта заданного типа.

Синтаксис:

```
Int64 CreateObjectByTypeDialog(Int32 aObjectTypeID);
Int64 CreateObjectByTypeDialog(IDBObjectType aObjectType);
Int64 CreateObjectByTypeDialog(Guid aObjectTypeGuid);
Int64 CreateObjectByTypeDialog(string aObjectTypeName);
```

```

Int64 CreateObjectByTypeDialog(Int32[] aObjectTypeIDs);
Int64 CreateObjectByTypeDialog(IDBObjectType[] aObjectTypes);
Int64 CreateObjectByTypeDialog(Guid[] aObjectTypeGuids);
Int64 CreateObjectByTypeDialog(string[] aObjectTypeNames);
Int64 CreateObjectByTypeDialog(Int32 aObjectTypeID, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectByTypeDialog(IDBObjectType aObjectType, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectByTypeDialog(Guid aObjectTypeGuid, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectByTypeDialog(string aObjectTypeName, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectByTypeDialog(Int32 aObjectTypeID, ObjectRelationLink[] aObjRelations, DateTime
aStartDate);
Int64 CreateObjectByTypeDialog(IDBObjectType aObjectType, ObjectRelationLink[] aObjRelations, DateTime
aStartDate);
Int64 CreateObjectByTypeDialog(Guid aObjectTypeGuid, ObjectRelationLink[] aObjRelations, DateTime
aStartDate);
Int64 CreateObjectByTypeDialog(string aObjectTypeName, ObjectRelationLink[] aObjRelations, DateTime
aStartDate);
Int64 CreateObjectByTypeDialog(Hashtable aObjectTypeIDRelationTypeIDs, Int64[] aRelatedObjectIDs);
Int64 CreateObjectByTypeDialog(Hashtable aObjectTypeIDRelationTypeIDs, Int64[] aRelatedObjectIDs,
DateTime aStartDate);

```

Параметры:

aObjectTypeID

Идентификатор типа объекта, по которому будет создан новый экземпляр объекта.

aObjectType

Тип объекта, по которому будет создан новый экземпляр объекта.

aObjectTypeGuid

Глобальный идентификатор типа объекта, по которому будет создан новый экземпляр объекта.

aObjectTypeName

Наименование типа объекта, по которому будет создан новый экземпляр объекта. Если aObjectTypeName не задано, то будет предложено выбрать тип объекта в диалоге.

aObjectTypeIDs

Массив идентификаторов типов объектов, из которых будет разрешено выбрать тип создаваемого экземпляра объекта.

aObjectTypes

Массив типов объектов, из которых будет разрешено выбрать тип создаваемого экземпляра объекта.

aObjectTypeGuids

Массив GUID типов объектов, из которых будет разрешено выбрать тип создаваемого экземпляра объекта.

aObjectTypeNames

Массив наименований типов объектов, из которых будет разрешено выбрать тип создаваемого экземпляра объекта.

aObjRelations

Массив структур, содержащих описание связей, которые нужно создать для нового объекта

aStartDate

Дата, с которой начинают действовать созданные связи.

aObjectTypeIDRelationTypeIDs

Хэш-таблица, в которой идентификатору типа объекта поставлен в соответствие идентификатор типа связей, которые нужно будет создать с объектами. Идентификаторы объектов переданы в aRelatedObjectIDs.

aRelatedObjectIDs

Массив идентификаторов, в состав которого нужно включить идентификатор создаваемого объекта.

Возвращаемое значение:

Идентификатор созданного объекта. Возвращается -1, если объект не создан.

CreateObjectDialog

Создание нового объекта.

Синтаксис:

```
Int64 CreateObjectDialog();
```

Возвращаемое значение:

Идентификатор созданного объекта.

CreateObjectVersionByTemplateDialog

Создание новой версии объекта по прототипу.

Синтаксис:

```
Int64 CreateObjectVersionByTemplateDialog(IDBObject aTemplateObject);
Int64 CreateObjectVersionByTemplateDialog(Int64 aTemplateObjectID);
Int64 CreateObjectVersionByTemplateDialog(Guid aObjectGuid);
Int64 CreateObjectVersionByTemplateDialog(IDBObject aTemplateObject, ObjectRelationLink[]
aObjRelations);
Int64 CreateObjectVersionByTemplateDialog(Int64 aTemplateObjectID, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectVersionByTemplateDialog(Guid aObjectGuid, ObjectRelationLink[] aObjRelations);
Int64 CreateObjectVersionByTemplateDialog(IDBObject aTemplateObject, ObjectRelationLink[]
aObjRelations, DateTime aStartDate);
Int64 CreateObjectVersionByTemplateDialog(Int64 aTemplateObjectID, ObjectRelationLink[] aObjRelations,
DateTime aStartDate);
Int64 CreateObjectVersionByTemplateDialog(Guid aObjectGuid, ObjectRelationLink[] aObjRelations,
DateTime aStartDate);
```

Параметры:

aTemplateObject

Прототип объекта, по которому будет создан новый экземпляр

aTemplateObjectID

Идентификатор, который задает объект-прототип для создаваемого экземпляра.

aObjectGuid

Глобальный идентификатор объекта, который является прототипом для создаваемого экземпляра.

aObjRelations

Массив структур, содержащих описание связей, которые нужно создать для нового объекта.

aObjRelations

Массив структур, содержащих описание связей, которые нужно создать для нового объекта.

aStartDate

Дата, с которой начинают действовать созданные связи.

Возвращаемое значение:

Идентификатор созданного объекта. Возвращается -1, если объект не создан.

RegisterCreatorCustomService

Регистрация мастера создания объектов определенного типа для переопределения имеющегося по умолчанию диалога.

Синтаксис:

```
void RegisterCreatorCustomService(Int32 aObjectTypeID, System.Type aCustomServiceType);
```

Параметры:

aObjectTypeID

Идентификатор типа объектов, для которого производится замена стандартного мастера создания объектов.

aCustomServiceType

Ссылка на тип объекта, реализующий интерфейс перекрытого диалога создания объекта определенного типа.

UnregisterCreatorCustomService

Разрегистрация мастера создания объектов определенного типа.

Синтаксис:

```
void UnregisterCreatorCustomService(Int32 aObjectTypeID, System.Type aCustomServiceType);
```

Параметры:

aObjectTypeID

Идентификатор типа объектов, для которого производится замена стандартного мастера создания объектов.

aCustomServiceType

Ссылка на тип объекта, реализующий интерфейс, который надо удалить из списка перекрывающих диалогов.

8.1.2 Замена стандартного мастера создания объектов

Для замены стандартного мастера создания объектов нужно создать класс, имеющий реализацию интерфейса `IObjectCreatorCustomService`. Этот интерфейс содержит метод:

```
Int64 CreateObjectDialog(Int32 aObjectTypeID),
```

который вызывает заменитель мастера создания объектов и возвращает идентификатор созданного экземпляра объекта, либо ноль в случае неудачи. В качестве параметра данный метод принимает идентификатор типа создаваемого объекта.

Пример реализации заменителя:

```
// диалог заменителя стандартного мастера создания объектов
public class MySampleForm: System.Windows.Forms.Form
{
    public int NewObjectID; // идентификатор созданного экземпляра объекта
    ...
}

// класс, реализующий службу для заменителя стандартного мастера создания объектов
public class MyObjectCreatorCustomService: IObjectCreatorCustomService
{
    private int _ObjectTypeID; // идентификатор типа объекта для которого реализован заменитель
    private IServiceProvider _isp;
    ...
    #region IObjectCreatorCustomService Members
    // вызов заменителя мастера создания объектов
    public long CreateObjectDialog(int aObjectTypeID)
    {
        MySampleForm sampleForm= new MySampleForm();
        if (sampleForm.ShowDialog() == DialogResult.OK)
            return sampleForm.NewObjectID;
        else
            return 0L;
    }
}
#endregion

// код внутри класса, производящего регистрацию и разрегистрацию службы
// для заменителя стандартного мастера создания объектов
...
Int32 _ObjectTypeID; // идентификатор типа объекта
...
#region Функции для регистрации и разрегистрации заменителя диалога
public void RegisterMyService()
{
    if (_isp != null)
    {
        IObjectCreatorService iobjCr =
        (IObjectCreatorService)_isp.GetService(typeof(IObjectCreatorService));
        if (iobjCr != null)
            iobjCr.RegisterCreatorCustomService(_ObjectTypeID, typeof(MyObjectCreatorCustomService));
    }
}

public void UnregisterMyService()
{
    if (_isp != null)
    {
        IObjectCreatorService iobjCr =
        (IObjectCreatorService)_isp.GetService(typeof(IObjectCreatorService));
        if (iobjCr != null)
            iobjCr.UnregisterCreatorCustomService(_ObjectTypeID, typeof(MyObjectCreatorCustomService));
    }
}
#endregion
...
```

Регистрацию (RegisterMyService) можно производить при инициализации модуля расширения (plug-in). Разрегистрацию (UnregisterMyService) нужно производить при выгрузке модуля расширения.

8.2 Контекстное меню

В клиентском приложении доступны два вида меню – общее, расположенное в главном окне приложения, и контекстные – доступные в элементах управления (гридах, дереве, т.п.). Программист имеет возможность добавлять в контекстные меню свои дополнительные команды, перекрывать или скрывать существующие.

Рассмотрим контекстные меню универсального клиента.

8.2.1 Добавление команд в контекстное меню

Для того, чтобы добавить или перекрыть команду в контекстном меню, программисту требуется выполнить несколько условий, а именно:

- реализовать в коде реализацию интерфейса ICommandsProvider,
- получить ссылку на интерфейс IFactory,
- при необходимости пополнить или изменить глобальный шаблон контекстного меню,
- зарегистрировать реализованный провайдер команд контекстного меню на определённую категорию и тип.

8.2.2 Интерфейс ICommandsProvider

Любой элемент управления, которому требуется поддержка контекстных меню, должен реализовать интерфейс ICommandsProvider.

8.2.2.1 Пространство имён

Intermech.Navigator.ContextMenu

Класс, реализующий данный интерфейс, является провайдером команд для контекстных меню. Перед отображением контекстного меню универсальный клиент опрашивает все зарегистрированные провайдеры, при этом передаёт им контекст – коллекцию выделенных элементов пространства навигации и контейнер сервисов для этих элементов. Провайдеры обязаны проанализировать полученный контекст и должны вернуть контейнер со списком допустимых команд для отображения в контекстном меню.

8.2.2.2 Методы интерфейса

GetMergedCommands

Метод вызывается для получения допустимых и подавляемых команд контекстного меню для выделенных элементов навигации одной категории и типа. Например, если в «Навигаторе» выделены элементы навигации нескольких разных категорий и типов, то данная команда будет вызываться для каждой из подгрупп этих элементов, сгруппированных по их категориям и типам. Наиболее применяемый метод данного интерфейса. Позволяет перекрывать команды контекстного меню для элементов навигации определённых категорий, типов, задавая более высокий приоритет описаниям этих команд.

Синтаксис:

```
CommandsInfo GetMergedCommands (ISelectedItems items, IServiceProvider viewServices);
```

Параметры:

items

Коллекция выделенных элементов пространства навигации, для которых выполняется поиск допустимых команд контекстного меню.

viewServices

Контейнер сервисов (контекст) для выделенных элементов пространства навигации.

Результат:

Контейнер со сведениями о допустимых и подавляемых командах контекстного меню.

Внимание! Основное требование к данному методу – нельзя выполнять обращения к базе данных для того, чтобы проверить, можно ли отображать команду меню или нет!

GetGroupCommands

Метод вызывается для получения допустимых и подавляемых команд контекстного меню для всей группы выделенных элементов навигации. Особенности данного метода:

1. Если команда зарегистрирована на все категории, то метод вызывается один раз и получает в качестве параметра `items` все выделенные в **Навигаторе** элементы навигации;
2. Если команда зарегистрирована на конкретную категорию, то метод будет вызван один раз для всех выделенных элементов навигации только в том случае, если все они принадлежат одной категории; для всех выделенных элементов навигации только в том случае, если все они принадлежат указанной категории;
3. Если команда зарегистрирована на конкретные категорию и тип, то метод будет вызван один раз для всех выделенных элементов навигации только в том случае, если все они принадлежат указанной категории и типу.

Синтаксис:

```
CommandsInfo GetGroupCommands (ISelectedItems items, IServiceProvider viewServices);
```

Параметры:

`items`

Коллекция выделенных элементов пространства навигации, для которых выполняется поиск допустимых команд контекстного меню.

`viewServices`

Контейнер сервисов (контекст) для выделенных элементов пространства навигации.

Результат:

Контейнер со сведениями о допустимых и подавляемых командах контекстного меню.

Внимание! Основное требование к данному методу – нельзя выполнять обращения к базе данных для того, чтобы проверить, можно ли отображать команду меню или нет!

8.2.3 Класс `CommandsInfo`

Класс `CommandsInfo` является контейнером сведений о допустимых и подавляемых командах контекстных меню. Экземпляры данного класса возвращаются в методах классов, реализующих интерфейс `ICommandsProvider`.

8.2.3.1 Пространство имён

`Intermech.Navigator.ContextMenu`

8.2.3.2 Статические свойства класса

`CommandsInfo.Empty { get; }`

Если провайдер команд контекстных меню должен вернуть пустую таблицу с командами, он может вернуть значение статического свойства `CommandsInfo.Empty`, вместо того, чтобы создавать и возвращать пустые экземпляры данного класса.

8.2.3.3 Свойства класса

`string[] CommandNames { get; }`

Возвращает массив имен допустимых команд. Если в контейнер не было добавлено ни одной такой команды, то результатом будет `null`.

8.2.3.4 Методы класса

Add

Метод добавляет в контейнер команду и соответствующий ей метод, появление которой в меню допустимо для данного контекста.

Синтаксис:

```
void Add (string commandName, CommandInfo commandInfo);
```

Параметры:

`commandName`

Уникальное имя добавляемой команды.

`commandInfo`

Информация о методе, реализующем выполнение данной команды после её выбора в меню.

Supress

Метод добавляет в контейнер команду, появление которой должно быть подавлено.

Синтаксис:

```
void Suppress (string commandName, Int32 priority);
```

Параметры:

commandName

Уникальное имя подавляемой команды.

priority

Приоритет подавляемой команды.

GetInfo

Возвращает информацию о методе, реализующем выполнение команды с указанным именем.

Синтаксис:

```
CommandInfo GetInfo (string commandName);
```

Параметры:

commandName

Уникальное имя команды контекстного меню.

Результат:

Информация о данной команде или null.

8.2.4 Класс CommandInfo

Класс описывает команду контекстного меню – её приоритет, делегаты для обработчика команды и для её отрисовки (если требуется). Экземпляры данного класса используются в коллекции CommandsInfo.

8.2.4.1 Пространство имён

Intermech.Navigator.ContextMenu

8.2.4.2 Свойства класса

```
Int32 Priority { get; }
```

Возвращает приоритет команды. Он используется сборщиком контекстного меню только в том случае, если несколько провайдеров одного уровня представляют одну и ту же команду. В этом случае будет выбрана команда с максимальным приоритетом.

```
ClickEventHandler ClickHandler { get; }
```

Возвращает делегат метода, который вызывается для выполнения команды контекстного меню.

```
DrawEventHandler DrawHandler { get; }
```

Возвращает делегат метода, который вызывается для отрисовки пункта контекстного меню, соответствующего команде.

```
object AdditionalInfo { get; }
```

Возвращает дополнительные сведения, которые должны быть переданы методу, вызываемому для выполнения команды контекстного меню.

8.2.5 Добавление элементов в шаблон контекстного меню

Контекстное меню универсального клиента строится на основе глобального шаблона – экземпляра класса MenuTemplate. Провайдеры команд контекстных меню управляют видимостью и реализацией этих команд, которые описаны в шаблоне. Следовательно, для того, чтобы команда появилась в контекстном меню, для начала требуется добавить её в шаблон. Рассмотрим это на примере. Добавим команду **Создать копию** в подменю **Создать**.

```
// Получаем ссылку на службу IFactory
IFactory factory = (IFactory)ServicesManager.GetService(typeof(IFactory));
// Получаем ссылку на глобальный шаблон контекстных меню
MenuTemplate template = factory.ContextMenuTemplate;
// Начинаем обновление шаблона
template.BeginUpdate();
// Найдём шаблон меню "Создать"
MenuTemplateNode createNode = factory.ContextMenuTemplate["Create"];
// Получаем ссылку на коллекцию именованных рисунков
_namedImageList = ServicesManager.GetService(typeof(INamedImageList)) as INamedImageList;
// Шаблон найден
if (createNode != null)
```



```

{
    // Создаём новый шаблон команды "CreateCopy", назначаем ей индекс изображения "imgCreate",
    // номер группы 20 и номер в группе 200
    createNode.Nodes.Add(new MenuTemplateNode("CreateCopy",
        "Создать копию объекта", _namedImageList.ImageIndex("imgCreate"), 20,
200));
}

```

В данном примере получают ссылки на службы IFactory и INamedImageList, затем получается ссылка на глобальный шаблон контекстных меню, в котором по имени отыскивается шаблон команды «Создать». И в итоге создаётся шаблон новой команды с именем **CreateCopy**, изображением «imgCreate» из общего списка именованных изображений. Каждой команде назначается номер группы и номер внутри группы. Согласно этим настройкам команды группируются и сортируются внутри шаблона.

8.2.6 Стандартные команды контекстного меню

В ядре универсального клиента уже есть стандартный шаблон контекстного меню, который затем расширяется за счёт плагинов. В приложении приведена таблица некоторых из этих команд. Программист может перекрывать эти команды в провайдерах, а также может добавлять в состав элементов контекстных меню этих команд свои вложенные меню.

8.2.7 Перекрывание существующих команд в контекстном меню

Перекрыть существующую команду в контекстном меню можно несколькими способами.

1. Если есть общая команда (не зарегистрирована на конкретную категорию или тип элементов навигации, то аналогичная команда, зарегистрированная на конкретную категорию, будет перекрывать общую команду;
2. Если есть команда, зарегистрированная на конкретную категорию элементов навигации, то аналогичная команда, зарегистрированная на эту категорию и конкретный тип, будет перекрывать как общую команду, так и команду на категорию;
3. Можно зарегистрировать аналогичную команду, но задать ей более высокий приоритет (см. класс TriggerPriority).

8.2.8 Приоритеты команд контекстных меню – класс TriggerPriority

Когда провайдер анализирует полученный контекст, он может определить, с каким приоритетом следует регистрировать команды для контекстного меню. Для этого используются константы класса TriggerPriority.

8.2.8.1 Пространство имен

Intermech.Navigator.Interfaces

8.2.8.2 Статические поля класса:

static Int32 Basic

Приоритет базовый - команда назначается при наличии выделенных элементов навигации.

static Int32 ItemCategory

Команда назначается после проверки категории выделенных элементов навигации.

static Int32 ItemData

Команда назначается после проверки данных выделенных элементов навигации.

static Int32 ItemType

Команда назначается после проверки типа выделенных элементов навигации.

static Int32 ParentCategory

Команда назначается после проверки категории родительских выделенных элементов навигации.

static Int32 ParentData

Команда назначается после проверки данных родительских выделенных элементов навигации.

static Int32 ParentType

Команда назначается после проверки типа родительских выделенных элементов навигации.

static Int32 ViewServices

Команда назначается после проверки контейнера сервисов (контекста выделенных элементов навигации).

В зависимости от того, что именно проверяется провайдером, определяет приоритет команды контекстного меню. Чем более специализирована команда, тем выше её приоритет. Константы класса `TriggerPriority` можно объединять с помощью логического **ИЛИ**, если провайдер выполняет сразу несколько проверок. Это ещё больше увеличивает приоритет команды.

8.2.9 Дополнительная настройка команд меню

Универсальный клиент предоставляет возможность настроить видимость команд контекстного меню для ролей, а затем скопировать эти настройки для пользователей. Причём настройки пользователей имеют более высокий приоритет перед настройками ролей. Данные настройки позволяют скрывать или разрешать отображение команд в контекстных меню.

Для этого используются экземпляры классов `AdjustableMenuCommand` и `AdjustableMenuCommands`, которые работают с глобальным шаблоном команд контекстного меню.

Получить доступ к текущей пользовательской настройке контекстных меню можно с помощью следующего примера:

```
// Получаем сервис настройки команд меню из системного контейнера сервисов
AdjustableMenuCommands menus = ServicesManager.GetService(typeof(AdjustableMenuCommands))
as AdjustableMenuCommands;
```

А затем можно воспользоваться методами и свойствами полученной коллекции.

Рассмотрим подробно классы, управляющие настройкой контекстных меню.

8.2.10 Класс `AdjustableMenuCommand`

Экземпляры класса `AdjustableMenuCommand` являются аналогами элементов шаблона контекстного меню, за исключением того, что в них есть дополнительные свойства, управляющие видимостью команд, а также свойства для построения иерархической структуры контекстного меню со всеми его подменю. По сути дела – это описания настраиваемых команд контекстных меню. На их основе можно построить шаблон контекстного меню или наоборот – на основе шаблона построить коллекцию описаний.

8.2.10.1 Пространство имён

`Intermech.Navigator.ContextMenu`

8.2.10.2 Свойства класса

`AdjustableMenuCommands Parent { get; }`

Родительская коллекция, которой принадлежит данное описание настраиваемой команды контекстного меню.

`AdjustableMenuCommands Items { get; }`

Коллекция дочерних команд.

`string Command { get; }`

Уникальная в пределах всей системы команда контекстного меню.

`string Caption { get; }`

Краткое текстовое описание команды.

`string Hint { get; }`

Более подробное текстовое описание команды.

`Int32 ImageIndex { get; }`

Индекс значка команды (из коллекции именованных значков).

`bool Visible { get; set; }`

Флажок позволяет прятать или показывать данную команду в контекстных меню "Навигатора".

`Int32 Group { get; }`

Номер группы команды.

`Int32 OrderBy { get; }`

Порядковый номер внутри группы.

`AdjustableMenuCommand this[Int32 index] { get; }`

Дочерняя команда с указанным индексом `index`.

`Int32 Count { get; }`

Количество дочерних команд в коллекции `Items`.

8.2.10.3 Конструктор

Синтаксис:

```
AdjustableMenuCommand (AdjustableMenuCommands parent, string command,  
string caption, bool visible, string hint, int imageIndex, int group, int orderID);
```

Параметры:

parent

Родительская коллекция, в состав которой будет входить вновь созданное описание команды контекстного меню.

command

Уникальная в пределах системы команда.

caption

Краткое текстовое описание команды.

visible

Показывать ли указанную команду в контекстных меню.

hint

Подробное текстовое описание команды.

imageIndex

Индекс изображения команды (в списке именованных значков).

group

Номер группы команды.

orderID

Порядковый номер в группе.

Результат:

Создаётся заполненный экземпляр класса.

Методы класса

Clone

Метод создаёт точную копию экземпляра класса.

Синтаксис:

```
object Clone ();
```

Результат:

Точная копия экземпляра класса.

FindCommand

Метод позволяет отыскать в коллекции команду с указанным именем. Поиск также будет проходить в дочерних коллекциях.

Синтаксис:

```
AdjustableMenuCommand FindCommand (string command);
```

Параметры:

command

Уникальное в пределах системы имя команды.

Результат:

Описание команды контекстного меню или null, если указанная команда не найдена.

Далее рассмотрим коллекцию AdjustableMenuCommands, на основе которой строится иерархия настраиваемых команд контекстного меню.

8.2.11 Класс AdjustableMenuCommands

С помощью данного класса программист может строить иерархические структуры настраиваемых контекстных меню. Рассмотрим свойства и методы данного класса. Класс унаследован от типизированного списка – List< AdjustableMenuCommand>. Поэтому у него доступны все свойства и методы базового класса, позволяющие управлять коллекцией дочерних элементов – описаний команд контекстного меню.

8.2.11.1 Пространство имён

Intermech.Navigator.ContextMenu

8.2.11.2 Свойства класса

```
AdjustableMenuCommands Parent { get; }
```

Родительская коллекция, которой принадлежит данная коллекция.

8.2.11.3 Конструкторы

Синтаксис:

```
AdjustableMenuCommands ();
```

Результат:

Создается пустой экземпляр класса.

Синтаксис:

```
AdjustableMenuCommands (AdjustableMenuCommands parent);
```

Параметры:

parent

Родительская коллекция, в состав которой будет входить вновь созданная коллекция.

Результат:

Создается заполненный экземпляр класса.

8.2.11.4 Методы класса

Assign

Метод выполняет полное присваивание данных из указанного источника.

Синтаксис:

```
void Assign (AdjustableMenuCommands source);
```

Параметры:

source

Источник, из которого будут получены данные в коллекцию.

Clone

Метод создает точную копию экземпляра класса.

Синтаксис:

```
object Clone ();
```

Результат:

Точная копия экземпляра класса.

FindCommand

Метод позволяет отыскать в коллекции команду с указанным именем. Поиск также будет проходить в дочерних коллекциях.

Синтаксис:

```
AdjustableMenuCommand FindCommand (string command);
```

Параметры:

command

Уникальное в пределах системы имя команды.

Результат:

Описание команды контекстного меню или null, если указанная команда не найдена.

FindCommandFromRoot

Метод позволяет отыскать в коллекции (начиная с её корневой родительской коллекции) команду с указанным именем. Поиск также будет проходить в дочерних коллекциях.

Синтаксис:

```
AdjustableMenuCommand FindCommand (string command);
```

Параметры:

command

Уникальное в пределах системы имя команды.

Результат:

Описание команды контекстного меню или null, если указанная команда не найдена.

8.3 Редактирование атрибутов объектов и связей

8.3.1 Класс ObjectPropertyGrid

Класс для редактирования атрибутов объектов и связей. Унаследован от класса PropertyGrid.

8.3.1.1 Свойства

LockTypeChange

При установке значения в *true* не дает изменять тип объекта. По умолчанию значение свойства *false*.

Синтаксис:

```
public bool LockTypeChange { get; set; }
```

8.3.1.2 Методы класса

Load

Инициализация компонента

Синтаксис:

```
public bool Load( Int64 aId, AttributableElements aAttributableElement, GetAttributeValuesModes aAttributeValuesModes, bool aIsChanged );
```

Параметры:

aId

Идентификатор элемента.

aAttributableElement

Тип элемента.

aAttributeValuesModes

Флаги отображения (см. enum GetAttributeValuesModes).

aIsChanged

После загрузки установить состояние изменения.

Save

Сохранение отредактированных атрибутов.

Синтаксис:

```
public bool Save();
```

8.4 Диалоги и формы вывода информации

8.4.1 Диалог для выбора метаданных

SelectorForm

Класс для диалога выбора.

Пространство имен: Intermech.PropertyEditors.

8.4.1.1 Свойства

IDList

Список идентификаторов выбранных объектов.

Синтаксис:

```
public ArrayList IDList { get { return idList; } }
```

NameList

Список выбранных наименований (соответствует списку idList).

Синтаксис:

```
public ArrayList NameList { get { return nameList; } }
```

TypeList

Список выбранных типов папок

Синтаксис:

```
public ArrayList TypeList { get { return typeList; } }
```

CategoryList

Список выбранных категорий папок.

Синтаксис:

```
public ArrayList CategoryList { get { return new ArrayList( FolderTypeArrayToCategoryArray( (Type[])typeList.ToArray( typeof(Type) ) )); } }
```

8.4.2 Диалог для выбора объектов

8.4.3 Класс SelectionWindow

Окно, позволяющее выбирать информационные объекты IPS Development Kit, связи и прочие объекты системы. Экземпляры данного окна кэшируются системой. В кэше допускается не более 15 таких окон.

Окно позволяет выполнить фильтрацию выбираемых объектов, т.е. запретить пользователю выбирать некоторые объекты. Для этих целей используются анализаторы `ISelectedItemsAnalyzer`, которые могут регистрироваться с помощью статического метода `RegisterAnalyzer`.

8.4.3.1 Пространство имен

`Intermech.Navigator`

8.4.3.2 События класса:

`Intermech.Navigator.Controls.SelectionWindowBeforeShow` `OnSelectionWindowBeforeShow`

Событие генерируется перед отображением формы по выбору объектов.

8.4.3.3 Статические методы класса:

`DynamicSelectObjects`

Выбрать объекты произвольного типа из базы данных (+ пользовательский обработчик событий).

Синтаксис:

```
static void DynamicSelectObjects (string caption; string description;  
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
Intermech.Interfaces.Client.DynamicSelectionEventHandler dynamicHandler;  
Intermech.Interfaces.Client.SelectionOptions options)  
static void DynamicSelectObjects (string caption; string description;  
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
Intermech.Interfaces.Client.DynamicSelectionEventHandler dynamicHandler;  
System.IServiceProvider nodesContext;  
Intermech.Interfaces.Client.SelectionOptions options)
```

Параметры:

`caption`

Заголовок.

`description`

Текст-пояснение.

`rootDescriptor`

Дескриптор корневого узла.

`dynamicHandler`

Пользовательский обработчик.

`options`

Внешний вид и поведение окна.

`nodesContext`

Контекст для узлов дерева и списков.

`RegisterAnalyzer`

Зарегистрировать анализатор для проверки выделенных элементов.

Синтаксис:

```
static bool RegisterAnalyzer (Intermech.Navigator.Controls.ISelectedItemsAnalyzer analyzer; bool  
temporary)
```

Параметры:

`analyzer`

Анализатор.

`temporary`

true - анализатор будет применяться только для одного окна выбора, после чего будет удалён из списка.

Возвращаемое значение:

true - анализатор был успешно добавлен, **false** - анализатор с таким Guid уже зарегистрирован в соответствующем списке (постоянных или временных анализаторов).

`UnregisterAnalyzer`

Удалить анализатор из списков.

Синтаксис:

```
static bool UnregisterAnalyze (Intermech.Navigator.Controls.ISelectedItemsAnalyzer analyzer)
```

Параметры:

analyzer

Анализатор.

Возвращаемое значение:

true - анализатор был успешно удалён, false - не был найден в списках.

Select

Выбрать объекты из базы данных на основании указанного дескриптора.

Синтаксис:

```
static object[] Select (string description;  
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
System.Type dataFormat;  
Intermech.Interfaces.Client.SelectionOptions options) static object[] Select (string description;  
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
System.Type dataFormat;  
System.IServiceProvider nodesContext;  
Intermech.Interfaces.Client.SelectionOptions options) static object[] Select (string caption;  
string description;  
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
System.Type dataFormat;  
Intermech.Interfaces.Client.SelectionOptions options) static object[] Select (string caption;  
string description; Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
System.Type dataFormat;  
System.IServiceProvider nodesContext;  
Intermech.Interfaces.Client.SelectionOptions options) static Int64[] SelectObjects (string caption;  
string description;  
Intermech.Interfaces.Client.SelectionOptions options) static Int64[] SelectObjects (string caption;  
string description;  
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
Intermech.Interfaces.Client.SelectionOptions options) static Int64[] SelectObjects (string caption;  
string description;  
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;  
System.IServiceProvider nodesContext;  
Intermech.Interfaces.Client.SelectionOptions options) static Int64[] SelectObjects (string caption;  
string description;  
Int32 objTypeID;  
Intermech.Interfaces.Client.SelectionOptions options) static Int64[] SelectObjects (string caption;  
string description;  
Int32 objTypeID;  
System.IServiceProvider nodesContext;  
Intermech.Interfaces.Client.SelectionOptions options) static Int64[] SelectObjects (string caption;  
string description;  
System.IServiceProvider nodesContext;  
Intermech.Interfaces.Client.SelectionOptions options)
```

Параметры:

caption

Заголовок.

description

Текст-пояснение.

rootDescriptor

Дескриптор корневого узла.

dataFormat

Формат данных.

options

Внешний вид и поведение окна.

nodesContext

Контекст для узлов дерева и списков.

objTypeID

Идентификатор типа объектов.

Возвращаемое значение:

Коллекция выбранных объектов.

Select

Выбрать объекты из базы данных на основании указанного дескриптора (+ заголовок окна, + обработка нажатия кнопки).

Синтаксис:

```
static object[] Select (string caption; string description;
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;
System.Type dataFormat;
Intermech.Interfaces.Client.DynamicSelectionEventHandler dynamicHandler;
Intermech.Interfaces.Client.SelectionOptions options) static object[] Select (string caption;
string description;
Intermech.Navigator.Interfaces.IDescriptor rootDescriptor;
System.Type dataFormat;
Intermech.Interfaces.Client.DynamicSelectionEventHandler dynamicHandler;
System.IServiceProvider nodesContext;
Intermech.Interfaces.Client.SelectionOptions options)
```

Параметры:

caption

Заголовок.

description

Текст-пояснение.

rootDescriptor

Дескриптор корневого узла.

dataFormat

Формат данных.

dynamicHandler

Обработчик нажатия кнопки.

nodesContext

Контекст для узлов дерева и списков.

options

Внешний вид и поведение окна.

Возвращаемое значение:

Коллекция выбранных объектов.

8.4.4 Перечислитель SelectionOptions

Набор флажков, позволяющий фильтровать выбираемые в окне SelectionWindow узлы (позволяет выбирать в окне узлы, содержащие определённые данные), скрывать и показывать дерево или панель закладок, управлять множественным выделением.

8.4.4.1 Пространство имен

Intermech.Interfaces.Client

8.4.4.2 Значения перечислителя:

Default = 0x1100

Позволяет выбирать любые узлы из дерева и списка, в окне доступны как дерево "Навигатора", так и панель закладок.

DisableMultiselect = 0x1000000

Запретить в гриде и дереве множественный выбор элементов, допускается выбор только одного элемента.

DisableObjectListFilter = 0x400000

Запретить фильтр списков объектов в окне.

DisableSelectAbstractTypes = 0x800000

Запретить в окне выбирать абстрактные типы объектов.

DisableSelectFromTree = 0x100000

Запретить выбирать в окне из дерева "Навигатора" (левая часть окна).

DisableSelectFromViews = 0x200000

Запретить выбирать в окне из закладок (правая часть окна).

ForceFilterObjectsByRule = 0x10000000

Принудительно включить кнопку "Фильтровать списки объектов по текущему правилу подбора версий".

ForceRebuildNavTree = 0x2000000

Принудительно вызвать перестроение дерева, даже если форма найдена в кэше.

HideTree = 0x2

Скрывать дерево "Навигатора" (левая часть окна).

HideViews = 0x1

Скрывать панель закладок (правая часть окна).

HideViewsGroupingBox = 0x20

Скрывать панель группирования в закладках.

HideViewsStatusBar = 0x40

Скрывать статусные строки в закладках.

HideViewsToolbar = 0x10

Скрывать панели управления в закладках.

SelectObjects = 0x100

Выбирать в окне можно узлы, содержащие объекты.

SelectObjectTypes = 0x200

Выбирать в окне можно узлы, содержащие типы объектов.

SelectOtherNodes = 0x10000

Выбирать в окне можно любые другие типы узлов.

SelectRelations = 0x1000

Выбирать в окне можно узлы, содержащие связи.

SelectRelationTypes = 0x2000

Выбирать в окне можно узлы, содержащие типы связей.

8.4.5 Перечислитель DynamicSelectionMode

Режим выбора объекта.

8.4.5.1 Пространство имен

Intermech.Interfaces.Client

8.4.5.2 Значения перечислителя:

PreSelect = 0x00000000

Производится проверка выбранного объекта.

Select = 0x00000001

Объект выбран и обработан ядром.

8.4.6 Делегат DynamicSelectionEventHandler

Делегат обработки динамического выбора из окна выбора Навигатора. Вызывается дважды для одного объекта. Сначала с режимом PreSelect, и, если возвращено true, второй раз после обработки ядром с режимом Select.

8.4.6.1 Пространство имен

Intermech.Interfaces.Client

8.4.6.2 Сигнатура

Синтаксис:

```
bool DynamicSelectionEventHandler (Int64 selectedObjectId;  
Intermech.Interfaces.Client.DynamicSelectionMode mode)
```

Параметры делегата:

selectedObjectId

Идентификатор выбранного объекта.

mode

Режим выбора объекта.

Возвращаемое значение:

Возвращает true для продолжения обработки объекта, false, если обработка объекта не нужна.

8.4.7 Делегат SelectionWindowBeforeShow

Событие генерируется перед отображением окна по выбору объектов.

8.4.7.1 Пространство имен

Intermech.Navigator.Controls

8.4.7.2 Сигнатура

Синтаксис:

```
void SelectionWindowBeforeShow (object sender; System.EventArgs e)
```

Параметры делегата:

sender

Отправитель (SelectionWindow).

e

Аргументы события.

8.4.8 Перечислитель SelectedItemsAnalyzerResult

Результат выполнения проверки коллекции выделенных элементов в ревизоре ISelectedItemAnalyzer.

8.4.8.1 Пространство имен

Intermech.Navigator.Controls

8.4.8.2 Значения перечислителя:

Disabled = 0x00000000

Элементы коллекции не являются подходящими, кнопка "ОК" становится запрещённой.

Enabled = 0x00000001

Элементы коллекции являются подходящими, кнопка "ОК" становится разрешённой.

8.4.9 Интерфейс ISelectedItemAnalyzer

Интерфейс фильтра, который выполняет анализ указанной коллекции ISelectedItem в окне по выбору объектов и возвращает значение, позволяющее заблокировать кнопку "ОК".

8.4.9.1 Пространство имен

Intermech.Navigator.Controls

8.4.9.2 Свойства интерфейса:

System.Guid Guid { get; }

Уникальный идентификатор анализатора.

8.4.9.3 Методы интерфейса:

Analyze

Выполнить анализирование указанной коллекции элементов, выделенных в окне.

Синтаксис:

```
Intermech.Navigator.Controls.SelectedItemAnalyzerResult Analyze  
(Intermech.Navigator.Controls.ISelectionWindow sender;  
Intermech.Navigator.Controls.ISelectedItemHost itemsHost)
```

Параметры:

sender

Окно, в котором осуществляется выбор элементов.

itemsHost

Служба окна, которая предоставляет коллекцию выделенных элементов.

Возвращаемое значение:

Результат проверки.

8.4.10 Класс ObjectTypesSelectedItemsAnalyzer

Анализатор разрешает кнопку "ОК" только в том случае, если выделены объекты допустимых типов (или их наследники).

8.4.10.1 Пространство имен

Intermech.Navigator.Controls

8.4.10.2 Реализуемые интерфейсы:

Intermech.Navigator.Controls.ISelectedItemsAnalyzer

8.4.10.3 Методы класса:

ObjectTypesSelectedItemsAnalyzer (конструктор)

Создать анализатор, добавить в список к выбору указанные типы объектов.

Синтаксис:

```
void ObjectTypesSelectedItemsAnalyzer (System.Collections.Generic.List<Int32> objectTypes; bool inheritance)
```

Параметры:

objectTypes

Разрешённые к выбору типы объектов.

inheritance

true - допускается выбирать объекты, тип которых унаследован от любого из указанных типов.

ObjectTypesSelectedItemsAnalyzer (конструктор)

Создать анализатор, добавить в список к выбору только указанный тип объекта.

Синтаксис:

```
void ObjectTypesSelectedItemsAnalyzer (Int32 objectType; bool inheritance)
```

Параметры:

objectType

Разрешённый к выбору тип объекта.

inheritance

true - допускается выбирать объекты, тип которых унаследован от указанного типа.

Analyze

Выполнить анализирование указанной коллекции элементов, выделенных в окне.

Синтаксис:

```
Intermech.Navigator.Controls.SelectedItemsAnalyzerResult Analyze  
(Intermech.Navigator.Controls.ISelectionWindow sender;  
Intermech.Navigator.Controls.ISelectedItemsHost itemsHost)
```

Параметры:

sender

Окно, в котором осуществляется выбор элементов.

itemsHost

Служба окна, которая предоставляет коллекцию выделенных элементов.

Возвращаемое значение:

Результат проверки.

8.4.11 Интерфейс IBackgroundTaskView

Сервис для отображения фоновых процессов в Навигаторе.

8.4.11.1 Пространство имен

Intermech.Interfaces.Client

8.4.11.2 Методы интерфейса:

AddTask

Добавить новое задание.

Синтаксис:

```
void AddTask (Intermech.Interfaces.Client.IBackgroundTask task)
```

Параметры:

task

Новое фоновое задание.

CheckClosing

Проверить, надо ли закрывать какое-либо фоновое задание.

Синтаксис:

```
bool CheckClosing ()
```

Возвращаемое значение:

false - нельзя остановить какое-то фоновое задание.

DeleteTask

Удалить фоновое задание.

Синтаксис:

```
void DeleteTask (Intermech.Interfaces.Client.IBackgroundTask task)
```

Параметры:

task

Удаляемое фоновое задание.

8.4.12 Интерфейс IBackgroundTask

Интерфейс фоновой задачи для представления в окне фоновых задач IBackgroundTaskView.

8.4.12.1 Пространство имен

Intermech.Interfaces.Client

8.4.12.2 Свойства интерфейса:

```
bool Active { get; }
```

Является ли активной указанная фоновая задача.

```
Int32 ImageIndex { get; }
```

Индекс изображения.

```
Int32 MaximumValue { get; set; }
```

Максимально допустимое значение индикатора прогресса.

```
Int32 MinimumValue { get; set; }
```

Минимально допустимое значение индикатора прогресса.

```
string Name { get; }
```

Название фоновой задачи.

```
object Result { get; set; }
```

Результат выполнения фоновой задачи.

```
Intermech.Interfaces.Client.BackgroundTaskShowMode ShowMode { get; }
```

Режим отображения состояния фоновой задачи.

```
Intermech.Interfaces.Client.BackgroundTaskState State { get; set; }
```

Текущее состояние фоновой задачи.

```
object Value { get; set; }
```

Текущее значение индикатора прогресса.

8.4.12.3 События интерфейса:

Intermech.Interfaces.Client.BackgroundTaskChangedEventHandler Changed

Событие, вызываемое при изменении состояния фоновой задачи.

8.4.12.4 Методы интерфейса:

CanPause

Можно ли приостанавливать фоновую задачу.

Синтаксис:

```
bool CanPause ()
```

Возвращаемое значение:

true - фоновую задачу можно приостанавливать.

CanResume

Можно ли возобновлять фоновую задачу.

Синтаксис:

```
bool CanResume ()
```

Возвращаемое значение:

true - фоновую задачу можно возобновлять.

CanStop

Можно ли останавливать фоновую задачу.

Синтаксис:

```
bool CanStop ()
```

Возвращаемое значение:

true - фоновую задачу можно останавливать.

CanTerminate

Можно ли прерывать фоновую задачу.

Синтаксис:

```
bool CanTerminate ()
```

Возвращаемое значение:

true - фоновую задачу можно прерывать.

Pause

Приостановить фоновую задачу.

Синтаксис:

```
void Pause ()
```

Resume

Возобновить фоновую задачу.

Синтаксис:

```
void Resume ()
```

SetMaxMin

Установить предельно допустимые значения для индикатора прогресса.

Синтаксис:

```
void SetMaxMin (Int32 max; Int32 min)
```

Параметры:

max

Максимально допустимое значение индикатора прогресса.

min

Минимально допустимое значение индикатора прогресса.

Stop

Остановить фоновую задачу.

Синтаксис:

```
void Stop ()
```

Terminate

Прервать фоновую задачу.

Синтаксис:

```
void Terminate ()
```

8.4.13 Перечислитель BackgroundTaskChangedType

Какие изменения произошли в фоновой задаче.

8.4.13.1 Пространство имен

Intermech.Interfaces.Client

8.4.13.2 Значения перечислителя:

All = 0x00000000

Любое из допустимых изменений.

Dispose = 0x00000006

Задача удаляется.

ImageIndex = 0x00000002

Изменился индекс изображения.

Result = 0x00000005

Изменился результат.

State = 0x00000004

Изменилось состояние.

Text = 0x00000001

Изменился текст.

Value = 0x00000003

Изменилось значение.

8.4.14 Перечислитель BackgroundTaskState

Текущее состояние фоновой задачи.

8.4.14.1 Пространство имен

Intermech.Interfaces.Client

8.4.14.2 Значения перечислителя:

Error = 0x00000004

Ошибка.

Paused = 0x00000001

Пауза.

Running = 0x00000000

Выполняется.

Stopped = 0x00000002

Остановлено.

Terminated = 0x00000003

Завершено.

8.4.15 Перечислитель BackgroundTaskShowMode

Режим отображения состояния фоновой задачи.

8.4.15.1 Пространство имен

Intermech.Interfaces.Client

8.4.15.2 Значения перечислителя:

Progress = 0x00000000

В виде индикатора прогресса.

Text = 0x00000001

В виде текстового сообщения.

8.4.16 Делегат BackgroundTaskChangedEventHandler

Делегат, вызываемый при изменении состояния фоновой задачи.

8.4.16.1 Пространство имен

Intermech.Interfaces.Client

8.4.16.2 Сигнатура

Синтаксис:

```
void BackgroundTaskChangedEventHandler (object sender;  
Intermech.Interfaces.Client.BackgroundTaskChangedType type)
```

Параметры делегата:

sender

Отправитель события.

type

Какие изменения произошли в фоновой задаче.

8.4.17 Интерфейс `IOutputView`

Служба позволяет отобразить текстовую информацию в окне вывода **Навигатора**.

8.4.17.1 *Пространство имен*

Intermech.Interfaces

8.4.17.2 *Методы интерфейса:*

Activate

Переключает окно вывода для заданной категории.

Синтаксис:

```
void Activate (string category)
```

Параметры:

category

Категория.

ClearText

Очищает окно вывода для указанной категории.

Синтаксис:

```
void ClearText (string category)
```

Параметры:

category

Категория.

ShowView

Показывает окно вывода.

Синтаксис:

```
void ShowView ()
```

WriteString

Выводит строку текста в окно вывода для заданной категории.

Синтаксис:

```
void WriteString (string category; string text)
```

Параметры:

category

Категория.

text

Выводимый текст.

8.4.18 Класс `ProgressForm`

Форма для отображения информации с индикатором прогресса.

8.4.18.1 *Пространство имен*

Intermech.PropertyEditors

8.4.18.2 *Поля класса:*

```
bool CanCloseForm
```

Можно ли закрыть форму.

8.4.18.3 *Свойства класса:*

```
Int32 Maximum { get; set; }
```

Максимальное значение для прогресс-бара.

```
Int32 ProgressValue { get; set; }
```

Значение для прогресс-бара.

8.4.18.4 *Методы класса:*

ProgressForm (конструктор)

Конструктор.

Синтаксис:

```
void ProgressForm (string FormCaption; string FormPromt; Int32 Value; Int32 ValueMax;  
bool ShowCancelButton; string CancelButtonCaption; System.EventHandler CancelButtonHandler)
```

Параметры:

FormCaption

Заголовок формы.

FormPromt

Поясняющий текст.

Value

Текущее значение прогресс-бара.

ValueMax

Максимальное значение прогресс-бара.

ShowCancelButton

true, если надо показать кнопку **Отмена**.

CancelButtonCaption

Заголовок кнопки **Отмена**.

CancelButtonHandler

Обработчик события для кнопки **Отмена**.

Dispose

Уберём за собой мусор.

Синтаксис:

```
void Dispose (bool disposing)
```

Параметры:

disposing

true if managed resources should be disposed; otherwise, false.

SetProgressValue

Установить новое значение для прогресс-бара.

Синтаксис:

```
void SetProgressValue (Int32 Value)
```

Параметры:

Value

Значение для прогресс-бара.

SetProgressValue

Установить новое значение для прогресс-бара, текст.

Синтаксис:

```
void SetProgressValue (Int32 Value; string text)
```

Параметры:

Value

Значение для прогресс-бара.

text

Текст.

8.4.18.5 Статические методы класса:

Execute

Статический метод для вызова формы. Создаёт, заполняет и показывает (с полной перерисовкой) форму.

Синтаксис:

```
static Intermech.PropertyEditors.ProgressForm Execute (string FormCaption;  
string FormPromt; Int32 Value; Int32 ValueMax;  
bool ShowCancelButton; string CancelButtonCaption; System.EventHandler CancelButtonHandler)
```

Параметры:

FormCaption

Заголовок формы.

FormPromt

Поясняющий текст.
 Value
 Текущее значение прогресс-бара.
 ValueMax
 Максимальное значение прогресс-бара.
 ShowCancelButton
 true, если надо показать кнопку **Отмена**.
 CancelButtonCaption
 Заголовок кнопки **Отмена**.
 CancelButtonHandler
 Обработчик события для кнопки **Отмена**.
 Возвращаемое значение:
 Вернёт ссылку на вновь созданную и показанную форму.

8.4.19 Класс IMessageBox

Окно для выдачи сообщения пользователю.

8.4.19.1 Пространство имен

Intermech.Controls

8.4.19.2 Статические методы класса:

Show

Отобразить окно с сообщением.

Синтаксис:

```
static System.Windows.Forms.DialogResult Show (string FormCaption;
string Message; Intermech.Controls.IMessageBoxButton Buttons)
static System.Windows.Forms.DialogResult Show (string FormCaption;
string Message; Intermech.Controls.IMessageBoxButton Buttons;
Intermech.Controls.IMessageBoxImage image)
static System.Windows.Forms.DialogResult Show (string FormCaption;
string Message; Intermech.Controls.IMessageBoxButton Buttons;
System.Drawing.Image image)
static Intermech.Controls.DialogResultAdv Show (string FormCaption;
string Message; Intermech.Controls.MessageBoxButtonsAdv messageBoxButtonsAdv;
Intermech.Controls.IMessageBoxImage image)
static Intermech.Controls.DialogResultAdv Show (string FormCaption;
string Message; Intermech.Controls.MessageBoxButtonsAdv messageBoxButtons; System.Drawing.Image image)
static System.Windows.Forms.DialogResult Show (string FormCaption;
string Message; System.Windows.Forms.MessageBoxButtons messageBoxButtons)
static System.Windows.Forms.DialogResult Show (string FormCaption;
string Message; System.Windows.Forms.MessageBoxButtons messageBoxButtons;
Intermech.Controls.IMessageBoxImage image)
static System.Windows.Forms.DialogResult Show (string FormCaption;
string Message; System.Windows.Forms.MessageBoxButtons messageBoxButtons; System.Drawing.Image image)
```

Параметры:

FormCaption

Заголовок окна.

Message

Сообщение.

Buttons

Кнопки.

messageBoxButtonsAdv

Кнопки.

image

Картинка / Тип картинки.

Возвращаемое значение:

Тип нажатой кнопки.

ShowEx

Отобразить окно с сообщением.

Синтаксис:

```
static object ShowEx (string FormCaption; string Message; Intermech.Controls.IMessageBoxButton Buttons)
static object ShowEx (string FormCaption; string Message; Intermech.Controls.IMessageBoxButton Buttons;
Intermech.Controls.IMessageBoxImage image)
static object ShowEx (string FormCaption; string Message; Intermech.Controls.IMessageBoxButton Buttons;
System.Drawing.Image image)
```

Параметры:

FormCaption

Заголовок окна.

Message

Сообщение.

Buttons

Кнопки.

image

Картинка.

Возвращаемое значение:

Tag нажатой кнопки.

8.4.20 Класс IMMessageBoxButton

Кнопка, размещаемая в окне IMMessageBox.

8.4.20.1 Пространство имен

Intermech.Controls

8.4.20.2 Поля класса:

string Caption

Надпись на кнопке.

System.Windows.Forms.DialogResult DialogResult

Возвращаемый кнопкой результат.

Intermech.Controls.DialogResultAdv DialogResultAdv

Возвращаемый кнопкой результат.

object Tag

Дополнительные данные.

8.4.20.3 Методы класса:

IMMessageBoxButton (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void IMMessageBoxButton (string caption;
Intermech.Controls.DialogResultAdv messageResult)
void IMMessageBoxButton (string caption; object tag)
void IMMessageBoxButton (string caption;
System.Windows.Forms.DialogResult messageResult)
void IMMessageBoxButton (string caption;
System.Windows.Forms.DialogResult messageResult; object tag)
```

Параметры:

caption

Заголовок кнопки.

messageResult

Результат кнопки.

tag

Дополнительные данные.

8.4.21 Перечислитель IMMessageBoxImage

Тип изображения для окна.

8.4.21.1 Пространство имен

Intermech.Controls

8.4.21.2 Значения перечислителя:

Error = 0x00000000
«Ошибка».
Information = 0x00000002
«Информация».
Question = 0x00000003
«Вопрос».
Warning = 0x00000001
«Предупреждение».

8.4.22 Перечислитель MessageBoxButtonsAdv

Расширенный набор кнопок для окна.

8.4.22.1 Пространство имен

Intermech.Controls

8.4.22.2 Значения перечислителя:

AbortRetryIgnore = 0x00000002
Окно содержит кнопки **Прервать, Повторить, Игнорировать**.
Ignore_IgnoreAll_Abort = 0x00000006
Окно содержит кнопки **Игнорировать, Игнорировать все**.
OK = 0x00000000
Окно содержит кнопку **ОК**.
OKCancel = 0x00000001
Окно содержит кнопку **Отмена**.
RetryCancel = 0x00000005
Окно содержит кнопки **Повторить, Отмена**.
YesNo = 0x00000004
Окно содержит кнопки **Да, Нет**.
YesNoCancel = 0x00000003
Окно содержит кнопки **Да, Нет, Отмена**.

8.5 Чтение и запись на сервер файлов и двоичных данных

Чтение и запись двоичных данных выполняется при помощи классов BlobProcReader и BlobProcWriter. При необходимости обработка данных может выполняться в отдельном фоновом потоке.

8.5.1 Класс BlobProcReader

Чтение двоичных данных из BLOB-полей СУБД напрямую и через потоки.

8.5.1.1 Пространство имен

Intermech.Client.Core

8.5.1.2 Свойства класса:

Intermech.Interfaces.BlobInformation BlobInformation { get; }
Структура с информацией о файле (BLOB-поле).
bool CloseDestStream { get; }
Закрывать поток-назначение.
System.IO.Stream DestStream { get; }
Поток, в который будет выполняться чтение данных из атрибута.

8.5.1.3 Методы класса:

BlobProcReader (конструктор)

Создать класс для чтения BLOB-поля.

Синтаксис:

```
void BlobProcReader (Intermech.Interfaces.IDBAttribute aIDBAttribute;  
Int32 aDataBlockSize;  
System.IO.Stream aDestStream; Intermech.Client.Core.BlobProcCustomClass.ProgressEventHandler aProgress;  
Intermech.Client.Core.BlobProcCustomClass.ThreadFinishEventHandler aThreadFinish)  
void BlobProcReader (Int64 aElementID;  
Intermech.AttributableElements aAttributableElement;  
Int32 aAttributeID;  
Int32 aIndex;  
Int32 aDataBlockSize;  
System.IO.Stream aDestStream; Intermech.Client.Core.BlobProcCustomClass.ProgressEventHandler aProgress;  
Intermech.Client.Core.BlobProcCustomClass.ThreadFinishEventHandler aThreadFinish)
```

Параметры:

aIDBAttribute

Интерфейс атрибута, из которого необходимо произвести чтение.

aElementID

Идентификатор объекта или связи.

aAttributableElement

Указывает, является значение, передаваемое в aElementID, идентификатором объекта или связи.

aAttributeID

Идентификатор атрибута, хранящего двоичные данные.

aIndex

Индекс значения (для многозначного атрибута).

aDataBlockSize

Размер блока зачитываемых данных. 0 – по умолчанию.

aDestStream

Поток, в который будет выполняться чтение данных из атрибута.

aProgress

Событие, сообщающее процент выполнения процесса чтения (0-100).

aThreadFinish

Событие, происходящее при завершении чтения данных при чтении данных в отдельном потоке.

ReadData

Произвести чтение информации из BLOB-поля.

Синтаксис:

```
void ReadData ()
```

ReadDataThread

Считывание двоичных данных из BLOB-поля в фоновом потоке. По окончании считывания данных вызывается обработчик, указанный в параметре конструктора aThreadFinish.

Синтаксис:

```
System.Threading.Thread ReadDataThread (bool aCloseDestStream)
```

Параметры:

aCloseDestStream

Указывает, производить ли закрытие потока aDestStream по окончании чтения данных.

Возвращаемое значение:

Фоновый поток, в котором выполняется чтение BLOB-поля.

8.5.2 Класс BlobProcWriter

Запись двоичных данных в BLOB-поля СУБД напрямую и через потоки.

8.5.2.1 Пространство имен

Intermech.Client.Core

8.5.2.2 Свойства класса:

`Intermech.Interfaces.BlobInformation BlobInformation { get; }`

Структура с информацией о файле (BLOB-поле).

`bool CloseDestStream { get; }`

Требуется ли закрывать поток-источник.

`System.IO.Stream SourceStream { get; }`

Поток, содержимое которого будет записано в BLOB-поле.

`System.IO.MemoryStream VirtualStream { get; }`

Поток, в который записывается данные вместо записи напрямую в базу данных. Запись в поток производится только при вызове метода `WriteData(true)`.

8.5.2.3 Методы класса:

BlobProcWriter (конструктор)

Создать класс для записи информации в BLOB-поле.

Синтаксис:

```
void BlobProcWriter (Intermech.Interfaces.IDBAttribute aIDBAttribute;  
Int32 aDataBlockSize; Intermech.Interfaces.BlobInformation aBlobInformation; System.IO.Stream  
aSourceStream;
```

```
Intermech.Client.Core.BlobProcCustomClass.ProgressEventHandler aProgress;
```

```
Intermech.Client.Core.BlobProcCustomClass.ThreadFinishEventHandler aThreadFinish)
```

```
void BlobProcWriter (Int64 aElementID;
```

```
Intermech.AttributableElements aAttributableElement;
```

```
Int32 aAttributeID;
```

```
Int32 aIndex;
```

```
Int32 aDataBlockSize;
```

```
Intermech.Interfaces.BlobInformation aBlobInformation; System.IO.Stream aSourceStream;
```

```
Intermech.Client.Core.BlobProcCustomClass.ProgressEventHandler aProgress;
```

```
Intermech.Client.Core.BlobProcCustomClass.ThreadFinishEventHandler aThreadFinish)
```

Параметры:

`aIDBAttribute`

Интерфейс атрибута, в который необходимо произвести запись.

`aElementID`

ID объекта/связи.

`aAttributableElement`

Признак объект/связь.

`aAttributeID`

Идентификатор атрибута.

`aIndex`

Индекс значения (для многозначного атрибута).

`aDataBlockSize`

Размер блока (0 - по умолчанию).

`aBlobInformation`

Описание BLOB-поля (не заполнять `BlobID`, `RealSize` и `PacketSize`; заполнять `Name`, `Note`, `ArcMethod`, `ModifyDate`).

`aSourceStream`

Поток-источник (содержимое будет записано в BLOB-поле).

`aProgress`

Событие, сообщающее процент выполнения процесса записи (0-100).

`aThreadFinish`

Событие, происходящее при завершении записи данных при записи данных в отдельном потоке.

CheckForZipExclusions

Если пишем файл и он подпадает под список исключений, то не выполнять упаковку данных.

Синтаксис:

```
void CheckForZipExclusions (Intermech.Interfaces.BlobInformation aBlobInformation)
```

Параметры:

aBlobInformation

Описание BLOB-поля.

WriteData

Записать информацию в BLOB-поле.

Синтаксис:

```
void WriteData ()
```

WriteData

Записать информацию в BLOB-поле.

Синтаксис:

```
void WriteData (bool virtualWrite)
```

```
void WriteData (bool virtualWrite; Intermech.Interfaces.IUserSession session)
```

Параметры:

virtualWrite

При значении true производится запись информации в virtualStream, а не в базу данных. Также производится заполнение BlobInformation.

session

Сессия.

WriteDataThread

Выполнить запись информации в BLOB-поле в фоновом потоке.

Синтаксис:

```
System.Threading.Thread WriteDataThread (bool aCloseSourceStream)
```

Параметры:

aCloseSourceStream

Закрывать ли поток при завершении процесса.

Возвращаемое значение:

Фоновый поток, в котором выполняется запись информации.

8.5.3 Перечислитель BlobProcessorMode

Состояние, в котором находится обработчик BLOB-поля.

8.5.3.1 Пространство имен

Intermech.Client.Core

8.5.3.2 Значения перечислителя:

Pack = 0x00000003

Упаковка.

Read = 0x00000001

Чтение.

Unknown = 0x00000000

Состояние неизвестно.

Unpack = 0x00000004

Распаковка.

Write = 0x00000002

Запись.

8.5.4 Примеры

Пример 1

```
using (SessionKeeper sk = new SessionKeeper())
{
    IDBObject iDBObject = sk.Session.GetObject(((long)12345));
    IDBAttributable iDBAttributable = iDBObject as IDBAttributable;
    IDBAttribute[] iDBAttributeCollection = iDBAttributable.Attributes.GetAttributesByType(
        FieldTypes.ftBlob );
    foreach ( IDBAttribute iDBAttribute in iDBAttributeCollection )
    {
        MemoryStream ms = new MemoryStream();
```

```

BlobProcReader bpr = new BlobProcReader( idBAttribute, 0, ms, null, null);
bpr.ReadData();
if (bpr.Result && ms.Length > 0)
{
    // ...
}
}

```

Пример 2

```

MemoryStream ms = new MemoryStream();
// заполнение ms
BlobInformation bi = new BlobInformation( 0, 0, DateTime.Now(), "1.bmp", ArcMethods.ZLibPacked,
"Изображение");
BlobProcWriter bpr = new BlobProcWriter((long)12345), AttributableElements.Object, (int)678, 0, 0, bi,
ms, null, null);
bpr.WriteData();

```

8.6 Добавление пунктов в общее дерево настроек системы

8.6.1 Интерфейс IPropertyPagesService

Сервис управления страницами свойств.

8.6.1.1 Пространство имен

Intermech.Interfaces.Client

8.6.1.2 События интерфейса:

System.EventHandler Changed

Событие возникает, когда на страничке свойств вносятся изменения в текущие настройки.

8.6.1.3 Методы интерфейса:

AddPage

Добавить страницу свойств.

Синтаксис:

```
void AddPage (string path; Intermech.Interfaces.Client.IPropertyPage page)
```

Параметры:

path

Путь и название страницы. В качестве разделителя используется '\'.

page

Страница свойств.

8.6.2 Интерфейс IPropertyPage

Элемент для отображения параметров настройки.

8.6.2.1 Пространство имен

Intermech.Interfaces.Client

8.6.2.2 Свойства интерфейса:

```
object Control { get; }
```

Объект для отображения свойств.

```
string HelpTopicID { get; }
```

id раздела справки для данного элемента управления.

```
string PageName { get; }
```

Имя страницы.

```
Intermech.Interfaces.Client.PropertyPageType Type { get; }
```

Тип страницы.

8.6.2.3 События интерфейса:

System.EventHandler Changed

Событие об изменении свойств на странице.

8.6.2.4 Методы интерфейса:

Apply

Сохранение изменений.

Синтаксис:

```
void Apply ()
```

Cancel

Отмена изменений.

Синтаксис:

```
void Cancel ()
```

8.6.3 Перечислитель PropertyPageType

Тип страницы свойств.

8.6.3.1 Пространство имен

Intermech.Interfaces.Client

8.6.3.2 Значения перечислителя:

Control = 0x00000001

Страничка представляет собой элемент, основанный на UserControl.

Object = 0x00000002

На страничке отображаются свойства объекта. Свойства объекта изменяются в PropertyGrid.

Unknown = 0x00000000

Не определен.

8.7 Обработка исключений

Исключительные ситуации (исключения, Exception) можно отображать в специальной форме ExceptionForm. В случае необработанных исключений клиентское приложение IPS Development Kit автоматически отобразит данную форму с информацией о возникшей ошибке.

Разработчикам настоятельно рекомендуется не подавлять возникающие исключения, а обрабатывать их, либо возвращать в систему с помощью throw.

Исключения, возникающие со стороны серверной части, и касающиеся прав доступа, автоматически отображаются в другой форме.

8.7.1 Класс ExceptionForm

Форма, предназначенная для отображения информации о возникшей исключительной ситуации (Exception). Для отображения следует использовать статический метод ShowExceptionDialog.

8.7.1.1 Пространство имен

Intermech.Client.Core

8.7.1.2 Статические методы класса:

ShowExceptionDialog

Отобразить информацию о возникшей исключительной ситуации (Exception).

Синтаксис:

```
static System.Windows.Forms.DialogResult ShowExceptionDialog (System.Exception e)
```

Параметры:

e

Возникшее исключение.

Возвращаемое значение:

Тип нажатой в окне кнопки.

8.8 Буфер обмена (Clipboard)

8.8.1 Интерфейс IClipboard

Интерфейс внутреннего буфера обмена клиентского приложения IPS Development Kit.

8.8.1.1 Пространство имен

Intermech.Interfaces.Client

8.8.1.2 События интерфейса:

`System.EventHandler Changed`

Событие возникает при изменении содержимого буфера обмена.

`System.EventHandler ContextChanged`

Событие возникает при изменении текущего объекта в буфере обмена.

8.8.1.3 Методы интерфейса:

GetDataObject

Возвращает текущий выделенный объект.

Синтаксис:

object GetDataObject ()

Возвращаемое значение:

Текущий выделенный объект.

GetDataObjects

Возвращает список объектов заданного типа.

Синтаксис:

object[] GetDataObjects ()

object[] GetDataObjects (System.Type needType)

Параметры:

`needType`

Требуемый тип объектов.

Возвращаемое значение:

Список объектов требуемого типа.

Pop

Восстанавливает ранее сохраненный набор объектов из временного массива.

Синтаксис:

void Pop ()

Push

Сохраняет набор объектов во временном массиве.

Синтаксис:

void Push ()

RefreshImage

Обновляет значок у выделенного объекта.

Синтаксис:

void RefreshImage ()

RemoveCurrentDataObject

Удалить из буфера обмена текущий объект.

Синтаксис:

void RemoveCurrentDataObject ()

RemoveDataObjects

Удаляет из буфера обмена объекты указанного типа.

Синтаксис:

void RemoveDataObjects (System.Type type)

Параметры:

type

Тип данных для удаления.

SetDataObject

Добавить информацию в буфер обмена.

Синтаксис:

```
void SetDataObject (object clipboardObject)
```

```
void SetDataObject (object clipboardObject; string title)
```

Параметры:

clipboardObject

Объект, добавляемый в буфер обмена.

title

Заголовок объекта.

8.8.2 Класс ClipboardObject

Объект, помещаемый в буфер обмена IPS Development Kit при операциях **Копировать** и **Вырезать**.

8.8.2.1 Пространство имен

Intermech.DataFormats

8.8.2.2 Реализуемые интерфейсы:

Intermech.DataFormats.IDBRelationID

Intermech.DataFormats.IDBTypedObjectID

8.8.2.3 Свойства класса:

```
Int64 BaseVersion { get; }
```

Признак базовой версии объекта.

```
string Caption { get; }
```

Заголовок объекта.

```
Int64 ID { get; }
```

Идентификатор объекта.

```
Intermech.DataFormats.IDBRelationID IDBRelationID { get; }
```

Интерфейс, описывающий связь, входящей в объект.

```
Intermech.DataFormats.IDBTypedObjectID IDBTypedObjectID { get; }
```

Интерфейс, описывающий пару тип объекта + его идентификатор.

```
Int64 ObjectID { get; }
```

Идентификатор версии объекта.

```
Int32 ObjectType { get; }
```

Идентификатор типа объекта.

```
Int64 Owner { get; }
```

Владелец объекта.

```
Int64 PartID { get; }
```

Идентификатор дочернего объекта.

```
Int32 RelationType { get; }
```

Идентификатор типа связи.

```
string SiteID { get; }
```

Узел информационной системы.

```
Int64 Sorting { get; }
```

Значение атрибута "Сортировка".

```
Int64 Value { get; }
```

Идентификатор связи.

```
Int64 Version { get; }
```

Номер версии объекта.

8.8.2.4 Методы класса:

ClipboardObject (конструктор)

Создать экземпляр класса.

Синтаксис:

```
void ClipboardObject (Intermech.DataFormats.IDBTypedObjectID idBTypedObjectID;  
Intermech.DataFormats.IDBRelationID idBRelationID)
```

Параметры:

idBTypedObjectID

Интерфейс, описывающий пару тип объекта + его идентификатор.

idBRelationID

Интерфейс, описывающий связь, входящей в объект.

ToString

Вернуть строковое описание объекта из буфера обмена.

Синтаксис:

```
string ToString ()
```

Возвращаемое значение:

Строковое описание объекта из буфера обмена.

8.8.3 Класс ClipboardObjectsList

Список объектов, помещённых в буфер обмена. Поддерживает интерфейс ICutCopy, чтобы иметь возможность отличать вырезанные объекты от скопированных.

8.8.3.1 Пространство имен

Intermech.Interfaces

8.8.3.2 Реализуемые интерфейсы:

Intermech.DataFormats.ICutCopy

Intermech.DataFormats.IDBObjectTypedIDCollection

Intermech.DataFormats.ITypedIDCollection

System.Collections.IEnumerator

8.8.3.3 Свойства класса:

```
Int32 ImageIndex { get; }
```

Индекс иконки.

```
bool IsCut { get; set; }
```

Признак того, что в буфер обмена помещены не скопированные, а вырезанные объекты.

8.8.3.4 Методы класса:

ClipboardObjectsList (конструктор)

Создать список объектов.

Синтаксис:

```
void ClipboardObjectsList (System.Collections.ArrayList idList; bool isCut)
```

Параметры:

idList

Список объектов.

isCut

true - объекты были помещены в список с помощью команды Вырезать.

ToString

Преобразование в строку.

Синтаксис:

```
string ToString ()
```

Возвращаемое значение:

Строковое представление того, что находится в буфере обмена.

8.8.4 Интерфейс ICutCopy

Интерфейс для передачи данных об типе операции (Вырезать, Копировать) при этих операциях через буфер обмена IPS Development Kit.

8.8.4.1 Пространство имен

Intermech.DataFormats

8.8.4.2 Свойства интерфейса:

Int32 ImageIndex { get; };

Индекс иконки в глобальной коллекции именованных значков NamedImageList.

bool IsCut { get; set; };

Флаг операции: true - объект вырезан, false - объект скопирован.

8.9 Вспомогательные классы и службы

8.9.1 Класс FormStorage

Класс для сохранения и восстановления позиции и размера модальных окон.

8.9.1.1 Пространство имен

Intermech.Client.Core

8.9.1.2 Статические методы класса:

LoadLayout

Загрузить состояние указанного элемента управления.

Синтаксис:

```
static void LoadLayout (System.Windows.Forms.Control control)
```

```
static void LoadLayout (System.Windows.Forms.Control control;
```

```
System.Collections.IDictionary iDictionary)
```

```
static void LoadLayout (System.Windows.Forms.Control control;
```

```
System.Collections.IDictionary iDictionary; bool returnDataOnly;
```

```
System.Drawing.Point lLocation; System.Drawing.Size lSize)
```

Параметры:

control

Элемент управления.

iDictionary

Дополнительные настройки для элемента управления.

returnDataOnly

true - только возврат данных, без присвоения значений в форму.

lLocation

Местонахождение.

lSize

Размеры.

SaveLayout

Сохранить настройки указанного элемента управления.

Синтаксис:

```
static void SaveLayout (System.Windows.Forms.Control control)
```

```
static void SaveLayout (System.Windows.Forms.Control control;
```

```
System.Collections.IDictionary iDictionary)
```

Параметры:

control

Элемент управления.

iDictionary

Дополнительные данные.

8.9.2 Регистрация расширений Навигатора – служба IFactory

Данная служба позволяет выполнять ряд действий, связанных с расширением возможностей основного компонента универсального клиента – **Навигатора**, а именно:

- регистрировать узлы **Навигатора** на определённые категории и типы,
- регистрировать провайдеры закладок **Навигатора** на определённые категории и типы,
- регистрировать/удалять провайдеры команд контекстных меню на определённые категории и типы,
- получать описания узлов определённой категории и типа,
- получать массивы провайдеров закладок **Навигатора** для определённых категорий и типов,
- получать массивы провайдеров команд контекстных меню для определённых категорий и типов,
- получать глобальный шаблон контекстного меню.

8.9.2.1 Пространство имён

Intermech.Navigator.Interfaces

Ссылка на службу, управляющую расширением возможностей «Навигатора». Её можно получить у общего контейнера сервисов – `ServicesManager` с помощью метода `GetService(typeof(IFactory))` as `IFactory`.

8.9.2.2 Свойства интерфейса

`MenuTemplate ContextMenuTemplate { get; }`

Ссылка на глобальный шаблон контекстного меню универсального клиента.

8.9.2.3 Методы интерфейса

AddNodeType

Метод регистрирует базовый тип элемента из пространства навигации для указанной категории и типа (при необходимости).

Синтаксис:

```
void AddNodeType (Int32 categoryID, Type nodeType);  
void AddNodeType (Int32 categoryID, Int32 typeID, Type nodeType);
```

Параметры:

`categoryID`

Идентификатор категории.

`typeID`

Идентификатор типа внутри категории.

`Type`

Регистрируемый тип элемента навигации.

AddViewsProvider

Метод регистрирует закладку «Навигатора», которая будет отображаться для элементов пространства навигации указанной категории и типа (при необходимости).

Синтаксис:

```
void AddViewsProvider (IViewsProvider provider);  
void AddViewsProvider (Int32 categoryID, IViewsProvider provider);  
void AddViewsProvider (Int32 categoryID, Int32 typeID, IViewsProvider provider);
```

Параметры:

`categoryID`

Идентификатор категории.

`typeID`

Идентификатор типа внутри категории.

`provider`

Регистрируемый провайдер закладок «Навигатора».

AddCommandsProvider

Метод регистрирует провайдер контекстных меню для элементов навигации указанной категории и типа (при необходимости).

Синтаксис:

```
void AddCommandsProvider (ICommandsProvider provider);  
void AddCommandsProvider (Int32 categoryID, ICommandsProvider provider);  
void AddCommandsProvider (Int32 categoryID, Int32 typeID, ICommandsProvider provider);
```

Параметры:

categoryID

Идентификатор категории.

typeID

Идентификатор типа внутри категории.

provider

Регистрируемый провайдер команд контекстных меню.

RemoveCommandsProvider

Метод удаляет указанный провайдер контекстных меню.

Синтаксис:

```
void RemoveCommandsProvider (ICommandsProvider provider);  
void RemoveCommandsProvider (Int32 categoryID, ICommandsProvider provider);  
void RemoveCommandsProvider (Int32 categoryID, Int32 typeID, ICommandsProvider provider);
```

Параметры:

categoryID

Идентификатор категории.

typeID

Идентификатор типа внутри категории.

provider

Удаляемый провайдер команд контекстных меню.

AddGlobalNode

Метод регистрирует элемент пространства навигации, который должен отображаться в составе корневого узла всей иерархии навигатора - **Информационное пространство**.

Синтаксис:

```
void AddGlobalNode (Guid descriptorGuid, IDescriptor descriptor, Int32 orderID);
```

Параметры:

descriptorGuid

Глобальный идентификатор дескриптора для регистрируемого элемента пространства навигации.

descriptor

Дескриптор регистрируемого элемента пространства навигации.

orderID

Порядковый номер регистрируемого элемента в составе корневого узла **Информационное пространство**.

GetNode

Метод создаёт и возвращает экземпляр элемента навигации для указанной категории и типа, либо согласно описанию этого элемента навигации

Синтаксис:

```
INode GetNode (Int32 categoryID, Int32 typeID);  
INode GetNode (INodeID nodeID, params object[] args);
```

Параметры:

categoryID

Идентификатор категории.

typeID

Идентификатор типа внутри категории.

nodeID

Описание элемента навигации.

args

Параметры, которые будут переданы в конструктор вновь создаваемому элементу навигации.

Результат:

Создаётся экземпляр элемента навигации для указанной категории и типа, либо по указанному описанию.

GetViewsProviders

Метод возвращает массив провайдеров закладок для элемента навигации указанной категории и типа (если необходимо). Если ни одного зарегистрированного провайдера найти не удалось, то метод возвращает null.

Синтаксис:

```
IViewsProvider[] GetViewsProviders ();  
IViewsProvider[] GetViewsProviders (Int32 categoryID);  
IViewsProvider[] GetViewsProviders (Int32 categoryID, Int32 typeID);
```

Параметры:

categoryID

Идентификатор категории.

typeID

Идентификатор типа внутри категории.

Результат:

Массив провайдеров закладок для элемента навигации указанной категории и типа (если необходимо), или null.

GetCommandsProviders

Метод возвращает массив провайдеров команд контекстного меню для элемента навигации указанной категории и типа (если необходимо). Если ни одного зарегистрированного провайдера найти не удалось, то метод возвращает null.

Синтаксис:

```
ICommandsProvider[] GetCommandsProviders ();  
ICommandsProvider[] GetCommandsProviders (Int32 categoryID);  
ICommandsProvider[] GetCommandsProviders (Int32 categoryID, Int32 typeID);
```

Параметры:

categoryID

Идентификатор категории.

typeID

Идентификатор типа внутри категории.

Результат:

Массив провайдеров контекстных меню для элемента навигации указанной категории и типа (если необходимо), или null.

8.9.3 Интерфейс ISimpleSelectedItem

Интерфейс предоставляет доступ к упрощённой коллекции выбранных пользователем элементов пространства навигации.

8.9.3.1 Пространство имен

Intermech.Navigator.Interfaces

8.9.3.2 Свойства интерфейса:

Int32 Count { get; }

Возвращает количество идентификаторов элементов навигации в коллекции.

8.9.3.3 Методы интерфейса:

GetItemData

Возвращает данные указанного формата для элемента коллекции. Если элемент не поддерживает указанный формат, то результатом будет null.

Синтаксис:

```
object GetItemData (Int32 index; System.Type dataFormat)
```

Параметры:

index

Индекс идентификатора элемента в коллекции.

dataFormat

Тип формата данных.

Возвращаемое значение:

Данные в указанном формате.

8.9.4 Интерфейс ISelectedItems

Интерфейс предоставляет доступ к коллекции выбранных пользователем элементов пространства навигации.

8.9.4.1 Пространство имен

Intermech.Navigator.Interfaces

8.9.4.2 Реализуемые интерфейсы:

Intermech.Navigator.Interfaces.ISimpleSelectedItems

8.9.4.3 Свойства интерфейса:

`bool IsCollage { get; }`

Возвращает true, если коллекция содержит разнородные идентификаторы элементов (т.е. созданные разными элементами навигации). Такие разнородные коллекции образуются при множественном выделении в дереве навигатора и других подобных этой ситуациях.

8.9.4.4 Методы интерфейса:

GetItemID

Возвращает идентификатор элемента в коллекции.

Синтаксис:

`Intermech.Navigator.Interfaces.INodeID GetItemID (Int32 index)`

Параметры:

`index`

Индекс идентификатора элемента в коллекции.

Возвращаемое значение:

Идентификатор элемента.

GetParentData

Возвращает данные требуемого формата для родительского элемента, создавшего указанный идентификатор элемента. Если родительский элемент не поддерживает запрошенный формат данных, то результатом будет null.

Синтаксис:

`object GetParentData (Int32 index; System.Type dataFormat)`

Параметры:

`index`

Индекс идентификатора элемента в коллекции.

`dataFormat`

Тип формата данных.

Возвращаемое значение:

Данные в указанном формате.

GetParentPath

Возвращает полный путь родительского элемента для указанного идентификатора в коллекции.

Синтаксис:

`Intermech.Navigator.Interfaces.NodeIDPath GetParentPath (Int32 index)`

Параметры:

`index`

Индекс идентификатора элемента в коллекции.

Возвращаемое значение:

Путь родительского элемента.

8.10 Отображение статусов объектов и связей

Каждый элемент (объект, связь), запрошенный клиентской стороной у серверной, может получать свои статусы в виде виртуального атрибута «Статусы элемента», который можно запросить у серверного ядра. Атрибут представляет собой битовый массив (физически передаётся как `byte[]`), где каждый бит или набор битов означают определённый статус.

Для управления этими наборами бит есть две службы – серверная и клиентская. У серверной службы регистрируются серверные плагины, которые будут добавлять свои статусы для элементов. При этом каждый такой плагин получает в своё распоряжение определённое «адресное пространство» в массиве бит статусов. Клиентская служба предназначена для корректной «раздачи» статусов клиентским плагинам, работающим в паре с серверными плагинами. Таким образом, серверный плагин, устанавливая в массиве бит свои статусы, не обязан знать, как физически будут размещены эти биты внутри массива. Клиентская сторона таким же образом, с помощью клиентского сервиса, получает свои статусы из общего массива, и также не обязана знать, как физически размещены эти биты.

Клиентская сторона может запрещать определённым серверным плагинам добавлять свои статусы. По умолчанию, любой серверный плагин добавляет свои статусы в результаты запроса, если в результирующей таблице найден столбец атрибута «Статусы элемента». Если в параметрах запроса данный атрибут не запрашивается, серверный плагин обязан не проводить обработку полученных данных и добавлять свои статусы.

8.10.1 Общие сведения.

Для работы механизма по управлению статусами используются следующие интерфейсы и классы:

`IElementStatusesService` – серверная служба, которая позволяет серверным расширениям регистрироваться у себя для корректной работы с битовыми массивами статусов элементов;

`ElementStatusesService` – класс, реализующий интерфейс `IElementStatusesService`. Класс реализован в серверном ядре системы;

`IElementStatusesClientService` – клиентская служба, которая после своей синхронизации с серверной службой `IElementStatusesService` позволяет клиентскому приложению считывать статусы элементов, установленные конкретными серверными плагинами;

`ElementStatusesClientService` – класс, который реализует интерфейс `IElementStatusesClientService`. Класс реализован в ядре клиентской части – в ядре «Навигатора»;

`ElementStatusesPluginDescription` – сериализуемый класс, с помощью экземпляров которого серверные плагины выполняют свою регистрацию у службы `IElementStatusesService`. Класс реализует интерфейс `IElementStatusesPluginDescription`;

`BitsArray` – статический класс, позволяющий выполнять ряд действий с массивом байт `byte[]`, представляя его в виде массива бит; используется серверной и клиентской службами по управлению статусами;

`PluginStatusesTable` – класс для передачи со стороны серверной службы на клиентскую таблицы `DataTable`, содержащей описания статусов, значки для статусов, от определённых серверных плагин.

Далее будет подробно рассмотрен механизм запроса статусов, их подготовки на серверной стороне, возврат на клиентскую сторону и обработка на клиентской стороне.

8.10.2 Запрос статусов со стороны клиентского приложения.

Запросить столбец со статусами можно, добавив в список запрашиваемых колонок `ColumnDescriptor[]` новый столбец, например, добавив в `ColumnDescriptor` следующий новый объект:

```
new ColumnDescriptor(ObligatoryObjectAttributes.F_ELEMENT_STATUSES, ColumnContents.Text,
    ColumnNameMapping.GUID, SortOrders.NONE, 0)
```

8.10.3 Запрет добавления статусов некоторым серверным плагинам.

Если необходимо запретить каким-либо серверным плагинам добавлять свои статусы, в поле `Tags` параметров запроса `DBRecordSetParams`, которое является гибридным словариком `HybridDictionary`, необходимо добавить пару значений `[(string)Запрещающий Guid плагина] = [(bool>true)]`, где запрещающий Guid можно получить у клиентской службы `IElementStatusesClientService`, зная Guid серверного плагина.

Например, так можно запретить плагину «Intermech.Pdm» добавлять свои статусы – описания связей как допустимые замены:

1. Создаём параметры запроса, в которые добавляем свои условия, список столбцов (включая столбец **Статусы элемента**):

```
DBRecordSetParams pars = new DBRecordSetParams(new ConditionStructure[] { .... } );
```

2. Добавляем в поле `Tags` свой ключик, при необходимости создавая саму коллекцию `Tags`:

```
if (pars.Tags == null) pars.Tags = new HybridDictionary(0, true);
pars.Tags[PDMPluginGuids.blockSubstitutesGuid] = true;
```

При этом серверная часть плагина проверяет наличие данного ключа и выключает свой механизм добавления статусов, даже несмотря на то, что столбец **Статусы элементов** был запрошен.

8.10.4 Регистрация серверного плагина для обработки статусов.

Каждый серверный плагин, которому требуется добавлять свои статусы в общие статусы элементов, должен выполнить некоторые действия. Рассмотрим на примере серверного плагина «Intermech.Pdm.Server», который должен при необходимости добавлять статусы в связи, указывая, являются ли связи допустимыми заменами.

1. Плагин создаёт в своём коде инициализации экземпляр класса `ElementStatusesPluginDescription`:

```
private ElementStatusesPluginDescription _pluginDescription = new ElementStatusesPluginDescription(4,
    PDMPPluginGuids.serverPdmGuid,
    PDMPPluginGuids.serverPdmGuidDisable, "Управление допустимыми заменами", "Статусы допустимых
заменителей");
```

Плагин резервирует под свои статусы 4 бита в общем массиве бит (значения от 0 до 15). Второй параметр является Guid-ом плагина, по которому серверная и клиентская службы могут отыскивать биты статусов. Третий параметр – запрещающий Guid; его можно задавать в коллекции Tags параметров запроса, чтобы запретить данному плагину добавлять свои статусы. Следующий параметр – название плагина. Последний параметр – название статусов, которые добавляются плагином в общий массив бит статусов.

2. Плагин запрашивает у менеджера сервисов службу `IPluginStatusesTable` и добавляет в неё описания своих статусов:

```
IPluginStatusesTable _pluginStatusesTable = serviceProvider.GetService(typeof(IPluginStatusesTable)) as
IPluginStatusesTable;
// Связь не участвует в допустимых заменах
_pluginStatusesTable.AddStatus(PDMPPluginGuids.serverPdmGuid,
    (Int32)RelationAsSubstitutes.rsNoSubstitutes,
    EnumDescConverter.GetEnumDescription(RelationAsSubstitutes.rsNoSubstitutes),
    null);
// Связь является актуальным заменителем
_pluginStatusesTable.AddStatus(PDMPPluginGuids.serverPdmGuid,
(Int32)RelationAsSubstitutes.rsActualSubstitute,
    EnumDescConverter.GetEnumDescription(RelationAsSubstitutes.rsActualSubstitute),
    LoadResource("Intermech.Pdm.Server.Resources.rsActualSubstitute.ico"));
```

Первый параметр метода `AddStatus` – это Guid плагина, второй параметр – значение статуса, для которого добавляются описание и изображение. В примере используются внутренние константы. Третий параметр – краткое описание статуса, которое будет показано в клиентском приложении. Четвёртый параметр – массив байт, содержащих изображение (иконку) для данного значения статуса. Значение `null` означает отсутствие значка. Метод `LoadResource` – внутренний метод, загружающий иконку из ресурсов плагина и возвращающий её в виде массива байт.

3. Плагин запрашивает у менеджера сервисов службу `IElementStatusesService` и регистрируется в ней:

```
IElementStatusesService _elementStatusesService =
    serviceProvider.GetService(typeof(IElementStatusesService)) as
    IElementStatusesService;
_elementStatusesService.RegisterServerPlugin(_pluginDescription);
```

После выполнения этих шагов серверная `IElementStatusesService` и клиентская службы `IElementStatusesClientService` по Guid плагина позволяют плагину корректно записывать или считывать статусы в массиве бит для обрабатываемых элементов (объектов, связей).

4. Серверный плагин обязан подписаться на событие `ILogHelper.BeforeRecordsSelectEvent` – для модификации параметров запроса непосредственно перед самим запросом (если плагину для установки статусов требуются дополнительные атрибуты, либо для проверки, запрещено ли плагину добавлять свои статусы).
5. Серверный плагин обязан подписаться на событие `ILogHelper.GetRecordsListEvent` – для обработки полученной после запроса таблицы, добавления своих статусов и для удаления из таблицы дополнительно запрошенных атрибутов.

8.10.5 Обработка статусов на серверной стороне.

На серверной стороне любой механизм, который обрабатывает данные после их запроса у коллекции объектов или связей, должен каким-то образом влиять на параметры запроса в базу данных, а также выполнить обработку полученной таблицы и назначить каждому элементу свои статусы.

Обработка статусов на серверной стороне происходит в два этапа.

1. Серверный плагин в обработчике `ILogHelper.BeforeRecordsSelectEvent` должен проверить `BeforeRecordsSelectEventArgs` `args.Tags` на наличие ключа со своим запрещающим `Guid`. И если он есть, то работа плагина по обработке статусов заканчивается.

Дополнительно плагин должен проверять, с какой коллекцией он работает. Например, плагин «Intermech.Pdm» работает с коллекциями связей, а при запросах в коллекции объектов он не должен выполнять какие-либо действия. Проверка типа коллекции в этом обработчике выглядит так:

```
// Попробуем выполнить приведение типов
DBRelationCollection rels = (sender as DBRelationCollection);
// На вход подана не коллекция связей - отваливаем
if (rels == null) return;
// Нет типа у коллекции связей - выход
if (rels.RelationTypeID == -1) return;
```

После всех проверок плагин может добавить в запрос дополнительные столбцы, если они требуются для назначения статусов запрашиваемых элементов.

2. В обработчике `ILogHelper.GetRecordsListEvent` серверный плагин получает таблицу элементов, которую он должен обработать. Если обработка элементов плагину запрещена (проверяется `BeforeRecordsSelectEventArgs` `parameters.Tags` на наличие ключа со своим запрещающим `Guid`), то никакая обработка не проводится.

Сначала отыскиваем индекс столбца атрибута «Статусы элемента» в таблице с помощью статического метода класса `ElementStatusesPluginDescription`:

```
int statusesIdx = ElementStatusesPluginDescription.GetStatusesColumnIndex(ref table);
Далее ищем сервис по управлению статусами элементов:
IElementStatusesService elementService =
    ServerServices.GetService(typeof(IElementStatusesService)) as IElementStatusesService;
```

Затем плагин должен обработать все строки таблицы, установить в каждой строке в столбце **Статусы элемента** соответствующий статус:

```
// Наш статус, сохранённый в Int16
Int16 State = Convert.ToInt16(status);
// Добавим его в столбец в массив бит
elementService.SetElementStatuses16(PDMPuginGuids.serverPdmGuid, dr[statusesIdx] as byte[], State);
```

Где `DataRow dr` – очередная строка в таблице, `status` – какое-то численное значение-статус.

После обработки плагин обязан удалить из таблицы все дополнительные столбцы, которые были добавлены этим плагином во время запроса в базу данных.

Таким образом, все подписавшиеся плагины добавят в столбец «Статусы элемента» свои статусы для каждого элемента.

8.10.6 Получение списка зарегистрированных серверных плагинов.

Клиентская сторона может получить список зарегистрировавшихся серверных плагинов следующим образом:

1. Получаем службу, управляющую статусами элементов:

```
IElementStatusesClientService _elementStatusesClientService =
    ServicesManager.GetService(typeof(IElementStatusesClientService)) as IElementStatusesClientService;
```

2. Получаем у этой службы гибридный словарь зарегистрированных плагинов:

```
// Получим ссылку на интерфейс перечислителя зарегистрированных плагинов
IDictionaryEnumerator myEnum = _elementStatusesClientService.Plugins.GetEnumerator();
// Вернём его внутренний указатель на начало списка
```

```

myEnum.Reset();
// Пройдём по всему списку зарегистрированных плагинов
while (myEnum.MoveNext())
{
    // Очередной зарегистрированный плагин
    ElementStatusesPluginDescription plugin = myEnum.Value as ElementStatusesPluginDescription;
    ...
}

```

Таким образом можно получить список Guid плагинов, список их запрещающих Guid, их названия и описание их статусов.

8.10.7 Получение статусов на клиентской стороне.

После вызова метода Select у коллекции объектов или связей клиентское приложение или клиентский плагин может для каждого элемента полученной таблицы извлекать статусы определённых плагинов.

1. Получаем службу, управляющую статусами элементов:

```

IElementStatusesClientService _elementStatusesClientService =
    ServicesManager.GetService(typeof(IElementStatusesClientService)) as IElementStatusesClientService;

```

2. Получаем статус элемента для любой строки из таблицы:

```

// Наш статус, установленный на сервере
Int16 State = elementStatusesClientService.GetElementStatuses16(
    PDMPPluginGuids.serverPdmGuid, dr[statusesIdx] as byte[]);

```

8.10.8 Получение описаний и изображений для статусов.

Серверная служба IPluginStatusesTable предоставляет механизм для регистрации описания статусов серверных плагинов, а также для чтения таблицы с этими описаниями на стороне клиента. Метод GetPluginStatusesTable позволяет запрашивать таблицу, в которую будут включены статусы с изображениями или без них. Кроме того, можно запросить описания и изображения только для определённых статусов.

8.11 Передача дополнительных параметров в запросах Select

Во время выполнения запросов к коллекциям объектов и связей может происходить модификация параметров запроса, а после выполнения запросов – фильтрация полученных таблиц с данными. Модификацию параметров запроса, а также фильтрацию данных могут выполнять как серверное ядро системы (например, механизм фильтрации состава – подбор версий объектов), так и серверные плагины, подписавшиеся на два события:

- IEventLogHelper.BeforeRecordsSelectEvent – изменение параметров запроса перед выполнением этого запроса в базе данных;
- IEventLogHelper.GetRecordsListEvent – фильтрация результирующей таблицы с данными.

Клиентская сторона может передавать в поле DBRecordSetParams.Tags дополнительные значения, которые будут интерпретироваться серверными плагинами и серверным ядром для выполнения дополнительных действий. Один из способов – непосредственное создание экземпляра этого класса и назначение его свойству DBRecordSetParams.Tags, а затем заполнение его сериализуемыми ключами и значениями. Пример приведён выше в п. [8.10.3](#).

Этот способ пригоден в том случае, если программист должен задать какие-то ключи только для конкретного запроса в базу данных. Если же есть необходимость задавать значения для всех запросов Select, вызываемых в окнах «Навигатора», следует воспользоваться службой Intermech.Interfaces.Client.IClientPluginsService.

8.11.1 Служба IClientPluginsService – дополнительные параметры запросов

8.11.1.1 Описание интерфейса

```
Intermech.Interfaces.Client.IClientPluginsService
```

Для сбора информации у клиентских плагинов и её передачи на серверную сторону можно воспользоваться службой Intermech.Interfaces.Client.IClientPluginsService. Данная служба позволяет регистрироваться плагинам (вернее, классам, реализующим интерфейс IClientPluginsDataTransfer). Перед запросом к коллекции объектов или связей данная служба вызывает у всех зарегистрированных интерфейсов метод GetPluginData

для сбора информации, а затем собранные данные помещает в поле DBRecordSetParams.Tags для отправки на серверную сторону.

На серверной стороне плагины могут извлекать информацию из этого поля и выполнять фильтрацию состава или списков объектов согласно полученных данных.

8.11.1.2 Свойства

SyncRoot

Объект для потокобезопасной работы со службой. Используется в основном для внутренних целей.

Синтаксис:

```
object SyncRoot { get; }
```

8.11.1.3 Методы

RegisterClientPlugin

Метод вызывается для регистрации в службе очередного плагина, реализующего интерфейс IClientPluginsDataTransfer.

Синтаксис:

```
void RegisterClientPlugin (Guid Plugin, IClientPluginsDataTransfer ClientPluginsDataTransfer);
```

Параметры:

Plugin

Уникальный Guid плагина, который надо зарегистрировать.

ClientPluginsDataTransfer

Ссылка на интерфейс плагина, методы которого будут вызываться в дальнейшем во время выполнения запросов к коллекциям объектов или связей.

UnregisterClientPlugin

Метод вызывается для исключения указанного плагина из службы.

Синтаксис:

```
void UnregisterClientPlugin (Guid Plugin);
```

Параметры:

Plugin

Уникальный Guid плагина, который надо исключить из службы.

GetClientPluginsData

Метод вызывается службой для сбора информации у плагинов. Плагины, подписавшиеся в коллекции IClientPluginsService, должны записать в словарь PluginsData свою информацию в виде сериализуемых пар значений [Ключ] = [Значение]. Указанная информация будет передана на серверную сторону во время вызова метода Select у коллекций объектов и связей в параметре DBRecordSetParams.Tags.

Синтаксис:

```
void GetClientPluginsData (ref HybridDictionary PluginsData);
```

Параметры:

PluginsData

Гибридный словарь, в котором в виде сериализуемых пар [ключ] = [значение] собирается информация, передаваемая от клиентских плагинов серверным.

PutClientPluginsData

Метод вызывается службой для раздачи информации плагинам. Плагины, подписавшиеся в коллекции IClientPluginsService, должны считать из словарика PluginsData свою информацию в виде сериализуемых пар значений [Ключ] = [Значение]. Метод PutPluginData зарезервирован для обратной связи серверной стороны с клиентской. В настоящее время не используется.

Синтаксис:

```
void PutClientPluginsData (ref HybridDictionary PluginsData);
```

Параметры:

PluginsData

Гибридный словарь, в котором в виде сериализуемых пар [ключ] = [значение] предоставляется информация для клиентских плагинов.

8.11.2 IClientPluginsDataTransfer – передача дополнительных параметров

8.11.2.1 Описание интерфейса

Intermech.Interfaces.Client.IClientPluginsDataTransfer

Механизм, позволяющий клиентским плагинам читать и записывать данные в коллекцию для передачи на серверную сторону во время вызова метода Select у коллекций объектов и связей.

8.11.2.2 Свойства

PluginGuid

Guid плагина, реализующего интерфейс IClientPluginsDataTransfer.

Синтаксис:

```
Guid PluginGuid { get; }
```

8.11.2.3 Методы

GetPluginData

Метод вызывается ядром клиентской части для сбора информации у плагинов. Плагины, подписавшиеся в коллекции IClientPluginsService, должны записать в словарь PluginsData свою информацию в виде сериализуемых пар значений [Ключ] = [Значение]. Указанная информация будет передана на серверную сторону во время вызова метода Select у коллекций объектов и связей в параметре DBRecordSetParams.Tags.

Синтаксис:

```
void GetPluginData (HybridDictionary PluginsData);
```

Параметры:

PluginsData

Гибридный словарь, в котором в виде сериализуемых пар [ключ] = [значение] собирается информация, передаваемая от клиентских плагинов серверным.

Плагин должен проверять значение данного параметра на значение null.

PutPluginData

Метод вызывается ядром клиентской части для раздачи информации плагинам. Плагины, подписавшиеся в коллекции IClientPluginsService, должны считать из словарика PluginsData свою информацию в виде сериализуемых пар значений [Ключ] = [Значение]. Метод PutPluginData зарезервирован для обратной связи серверной стороны с клиентской. В настоящее время не используется.

Синтаксис:

```
void PutPluginData (HybridDictionary PluginsData);
```

Параметры:

PluginsData

Гибридный словарь, в котором в виде сериализуемых пар [ключ] = [значение] предоставляется информация для клиентских плагинов.

Плагин должен проверять значение данного параметра на значение null.

8.12 Механизм отображения и сортировки составов в «Навигаторе»

При получении составов с сервера приложений клиентская часть может запросить определённую сортировку столбцов состава – в зависимости от того, по каким колонкам в дереве или в гриде задал сортировку пользователь. При этом состав может быть получен по типу связи по умолчанию, либо по другим типам связей (по одной или по нескольким). Кроме того, состав можно отсортировать по возрастанию значений атрибута связей «Сортировка». В этом случае сортировка состава называется ручной.

Настройки о том, как отображать составы (по каким типам связей и в каком порядке), а также о том, как выполнять автоматическое заполнение значений атрибута **Сортировка** (режим автоматической сортировки) хранятся в экземплярах объектов базы данных типа **Конфигурации ролей** (идентификатор типа объекта равен **SystemGUIDs.objtypeRoleConfigData**) в атрибуте **Сортировка и отображение составов** (идентификатор типа атрибута равен **SystemGUIDs.attributeKernelCompositionSorting**). Настроенные объекты применяются для ролей. У каждого объекта типа **Роль** есть ссылочный атрибут типа **Конфигурация роли** (**SystemGUIDs.attributeKernelRolesConfig**), который ссылается на настроенный объект.

Правила загружаются из объектов базы данных в экземпляры классов **CompositionsAutosortRule**.

Для получения текущего правила сортировки и отображения составов можно воспользоваться сервисом **ICurrentUserAndRole**, который можно получить у статического класса **ServicesManager**.

Рассмотрим подробно все классы, которые участвуют в сортировке и отображении составов.

8.12.1.1 Примечание

Все рассмотренные ниже классы поддерживают интерфейсы **ICloneable**, **IComparable** и **IComparable<>**.

8.12.2 Класс **CompositionsAutosortRule**

8.12.2.1 Пространство имён

Intermech.Interfaces

В экземплярах данного класса хранятся правила сортировки и отображения составов, а именно список родительских типов объектов, для которых сделаны настройки, списки допустимых типов связей для каждого из данных родительских типов объектов, а также списки допустимых дочерних типов объектов для каждого типа связи. Порядок допустимых типов связей играет важную роль, так как именно в этом порядке загружаются части разнородных составов. Часть типов связей может быть отключена, в этом случае составы по этим типам связей не будут отображаться в клиентском приложении. Порядок допустимых дочерних типов объектов важен при выполнении операции автоматической сортировки составов.

8.12.2.2 Свойства класса

`Int64 ObjectID { get; set; }`

Идентификатор версии объекта базы данных типа «Конфигурации ролей», из которого было загружено данное правило.

`Guid Guid { get; set; }`

Guid идентификатор версии объекта базы данных типа «Конфигурации ролей», из которого было загружено данное правило.

`string Name { get; set; }`

Заголовок версии объекта базы данных типа «Конфигурации ролей», из которого было загружено данное правило. Внимание. При изменении данного поля заголовок у самого объекта в базе данных не меняется.

`List<ParentObjectType> ParentObjectTypes { get; }`

Список родительских типов объектов, составы которых будут сортироваться и отображаться способом, отличным от стандартного.

8.12.2.3 Конструкторы

Синтаксис:

`CompositionsAutosortRule ();`

Результат:

Создаётся пустой экземпляр класса.

Синтаксис:

`CompositionsAutosortRule (Int64 objectID);`

Параметры:

`objectID`

Идентификатор версии объекта типа «Конфигурации ролей».

Результат:

Создаётся пустой экземпляр класса, у которого заполняется свойство **ObjectID**.

Синтаксис:

`CompositionsAutosortRule (Int64 objectID, Guid guid, string name, List<ParentObjectType> parentObjectTypes);`

Параметры:

`objectID`

Идентификатор версии объекта типа «Конфигурации ролей».

`guid`

Guid идентификатор версии объекта типа «Конфигурации ролей».

`name`

Заголовок версии объекта типа «Конфигурации ролей».

`parentObjectTypes`

Список родительских типов объектов.

Результат:

Создаётся экземпляр класса, у которого заполняются все свойства.

8.12.2.4 Методы класса

Assign

Метод позволяет скопировать информацию из другого правила в текущее.

Синтаксис:

void Assign(object source);

Параметры:

source

Объект, совместимый с типом **CompositionsAutosortRule**.

Clear

Метод позволяет очистить внутренние коллекции правила. При этом не очищаются идентификатор версии объекта, Guid и название правила.

Синтаксис:

void Clear();

Clone

Метод создаёт точную копию экземпляра класса.

Синтаксис:

object Clone ();

Результат:

Точная копия экземпляра класса.

GenerateStartSortingValues

Метод позволяет регенерировать заново стартовые значения атрибута «Сортировка» у всей коллекции родительских типов объектов.

Синтаксис:

void GenerateStartSortingValues ();

GetObjectTypeVisibleRelations

Метод позволяет получить список идентификаторов типов видимых связей для указанного родительского типа объектов.

Синтаксис:

List<Int32> GetObjectTypeVisibleRelations (Int32 ObjectType);

List<Int32> GetObjectTypeVisibleRelations (Guid ObjectTypeGuid);

Параметры:

ObjectType

Int32-идентификатор родительского типа объектов.

ObjectTypeGuid

Guid идентификатор родительского типа объектов.

Результат:

Список идентификаторов типов видимых связей для указанного родительского типа объектов.

GetObjectTypeVisibleRelationsGuids

Метод позволяет получить список Guid идентификаторов типов видимых связей для указанного родительского типа объектов.

Синтаксис:

List<Guid> GetObjectTypeVisibleRelationsGuids (Int32 ObjectType);

List<Guid> GetObjectTypeVisibleRelationsGuids (Guid ObjectTypeGuid);

Параметры:

ObjectType

Int32-идентификатор родительского типа объектов.

ObjectTypeGuid

Guid идентификатор родительского типа объектов.

Результат:

Список Guid идентификаторов типов видимых связей для указанного родительского типа объектов.

IndexOfParentObjectType

Метод позволяет получить номер указанного родительского типа объектов в коллекции.

Синтаксис:

Int32 IndexOfParentObjectType (Int32 parObjTypeID);

Параметры:

parObjTypeID

Int32-идентификатор родительского типа объектов.

Результат:

Номер указанного родительского типа объектов в коллекции или **-1**, если данный тип объекта не найден.

Load

Метод позволяет загрузить правило из указанного объекта базы данных.

Синтаксис:

void Load (IUserSession session, Int64 objectID, bool throwException);

Параметры:

session

Сессия, в рамках которой выполняется работа с базой данных.

objectID

Идентификатор версии объекта типа «Конфигурации ролей», из которого требуется выполнить загрузку всех свойств правила.

throwException

Если установить значение в **true**, то любое возникшее исключение будет возвращено в приложение. Иначе все исключения обрабатываются и подавляются.

Load

Метод позволяет загрузить правило из указанного XML-документа.

Синтаксис:

void Load (XMLSettingsStorage storage, XmlNode node);

Параметры:

storage

Хранилище XML-документа, из которого требуется загрузить все данные правила.

node

Узел XML, в котором хранятся свойства правила.

Save

Метод позволяет сохранить правило в указанный объект базы данных.

Синтаксис:

void Load (IUserSession session, Int64 objectID, bool throwException);

Параметры:

session

Сессия, в рамках которой выполняется работа с базой данных.

objectID

Идентификатор версии объекта типа «Конфигурации ролей», в который требуется сохранить все свойства правила.

throwException

Если установить значение в **true**, то любое возникшее исключение будет возвращено в приложение. Иначе все исключения обрабатываются и подавляются.

Save

Метод позволяет сохранить правило в указанный XML-документ.

Синтаксис:

```
void Save (XMLSettingsStorage storage, XmlNode node);
```

Параметры:

storage

Хранилище XML-документа, в которое требуется сохранить все данные правила.

node

Узел XML, в котором будут храниться свойства правила.

SyncMetadata

Метод выполняет полную синхронизацию содержимого экземпляра класса с системным кэшем метаданных. При этом будут обновлены списки родительских и дочерних типов объектов, а также допустимых типов связей, устранены любые различия между свойствами правила и системными метаданными.

Синтаксис:

```
void Load (IUserSession session);
```

Параметры:

session

Сессия, в рамках которой выполняется работа с базой данных.

8.12.3 Класс ParentObjectType

8.12.3.1 Пространство имён

Intermech.Interfaces

В экземплярах данного класса хранятся идентификатор родительского типа объектов, а также списки допустимых типов связей для каждого из данных родительских типов объектов, а также списки допустимых дочерних типов объектов для каждого типа связи. Порядок допустимых типов связей играет важную роль, так как именно в этом порядке загружаются части разнородных составов. Часть типов связей может быть отключена, в этом случае составы по этим типам связей не будут отображаться в клиентском приложении. Порядок допустимых дочерних типов объектов важен при выполнении операции автоматической сортировки составов.

8.12.3.2 Свойства класса

```
Int32 ObjectTypeID { get; set; }
```

Идентификатор родительского типа объектов, для которого хранятся настройки видимых типов связей, а также допустимые дочерние типы объектов.

```
List<ChildRelationType> ChildRelationTypes { get; }
```

Список допустимых типов связей. Каждый тип связи может быть включен или выключен. Также роль играет порядок типов связей в данном списке.

```
ChildRelationType this[Int32 relTypeID] { get; }
```

Описание допустимого типа связи с указанным Int32-идентификатором relTypeID или null, если указанный тип связи не найден.

8.12.3.3 Конструкторы

Синтаксис:

```
ParentObjectType ();
```

Результат:

Создается пустой экземпляр класса.

Синтаксис:

```
ParentObjectType (Int32 objectTypeID);
```

Параметры:

objectTypeID

Идентификатор родительского типа объектов.

Результат:

Создается пустой экземпляр класса, у которого заполняется свойство **ObjectTypeID**.

8.12.3.4 Методы класса

Assign

Метод позволяет скопировать информацию из другого родительского типа объекта в текущий.

Синтаксис:

```
void Assign(object source);
```

Параметры:

source

Объект, совместимый с типом **ParentObjectType**.

Clear

Метод позволяет очистить внутренние коллекции. При этом не очищается идентификатор родительского типа объекта.

Синтаксис:

```
void Clear();
```

Clone

Метод создаёт точную копию описания родительского типа объектов.

Синтаксис:

```
object Clone ();
```

Результат:

Точная копия описания родительского типа объектов.

GenerateStartSortingValues

Метод позволяет регенерировать заново стартовые значения атрибута «Сортировка» у всей коллекции допустимых типов связей.

Синтаксис:

```
void GenerateStartSortingValues ();
```

Load

Метод позволяет загрузить описание родительского типа объектов из указанного XML-документа.

Синтаксис:

```
void Load (XMLSettingsStorage storage, XmlNode node);
```

Параметры:

storage

Хранилище XML-документа, из которого требуется загрузить описание родительского типа объектов.

node

Узел XML, в котором хранятся свойства описания родительского типа объектов.

Save

Метод позволяет сохранить описание родительского типа объектов в указанный XML-документ.

Синтаксис:

```
void Save (XMLSettingsStorage storage, XmlNode node);
```

Параметры:

storage

Хранилище XML-документа, в которое требуется сохранить описание родительского типа объектов.

node

Узел XML, в котором требуется сохранить свойства описания родительского типа объектов.

SyncMetadata

Метод выполняет полную синхронизацию содержимого экземпляра класса с системным кэшем метаданных. При этом будут обновлены списки дочерних типов объектов, а также допустимых типов связей, устранены любые различия между свойствами правила и системными метаданными.

Синтаксис:

```
void Load (IUserSession session);
```

Параметры:

session

Сессия, в рамках которой выполняется работа с базой данных.

8.12.4 Класс ChildRelationType

8.12.4.1 Пространство имён

Intermech.Interfaces

В экземплярах данного класса хранятся идентификатор допустимого типа связи, настройка видимости данного типа связи, ссылка на родительский тип объекта, а также список допустимых дочерних типов объектов. Порядок допустимых типов дочерних объектов играет важную роль при выполнении операции автоматической сортировки составов.

8.12.4.2 Свойства класса

`Int32 RelationTypeID { get; set; }`

Идентификатор допустимого типа связи, для которого хранятся настройки допустимых дочерних типов объектов.

`Int32 ParentObjectTypeID { get; set; }`

Идентификатор родительского типа объектов, состав которого получается указанным типом связи.

`bool Visible { get; set; }`

Отображаются ли связи указанного типа в составах на клиентском приложении.

`List<ChildObjectType> ChildObjectTypes { get; }`

Список с описаниями допустимых дочерних типов объектов в составах. Согласно данному списку происходит автоматическая сортировка составов.

`ChildObjectType this[Int32 objTypeID] { get; }`

Получить описание допустимого дочернего типа объектов по его идентификатору. Если указанный тип не найден, будет возвращено значение null.

8.12.4.3 Конструкторы

Синтаксис:

`ChildRelationType ();`

Результат:

Создается пустой экземпляр класса.

Синтаксис:

`ChildRelationType (Int32 relationTypeID, Int32 parentObjectTypeID, bool visible);`

Параметры:

`relationTypeID`

Идентификатор допустимого типа связи.

`parentObjectTypeID`

Идентификатор родительского типа объектов, для которого создано описание допустимого типа связи.

`visible`

Управление видимостью данного типа связи в составах.

Результат:

Создается экземпляр класса, у которого заполняется все свойства.

8.12.4.4 Методы класса

Assign

Метод позволяет скопировать информацию из другого описания допустимого типа связи в текущий.

Синтаксис:

`void Assign(object source);`

Параметры:

`source`

Объект, совместимый с типом **`ChildRelationType`**.

Clear

Метод позволяет очистить внутренние коллекции и свойства. При этом не очищаются идентификаторы родительского типа объекта и типа связи.

Синтаксис:

`void Clear();`

Clone

Метод создаёт точную копию описания допустимого типа связи.

Синтаксис:

object Clone ();

Результат:

Точная копия описания допустимого типа связи.

GenerateStartSortingValues

Метод позволяет регенерировать заново стартовые значения атрибута «Сортировка» у всей коллекции дочерних типов объектов.

Синтаксис:

```
void GenerateStartSortingValues ();
```

GetNearestBaseParentObjectType

В списке допустимых типов объектов хранятся только первый уровень объектов. Если требуется отыскать позицию в списке для какого-либо дочернего типа объектов, который является наследником от одного из объектов из коллекции **ChildObjectTypes**, следует воспользоваться данным методом.

Синтаксис:

```
Int32 GetNearestBaseParentObjectType (Int32 childObjectType);
```

Параметры:

childObjectType

Идентификатор дочернего типа объекта, который находится в коллекции **ChildObjectTypes**, либо является наследником какого-либо из этих объектов.

Результат:

Порядковый номер указанного типа объекта или **-1**, если такой тип (или его родительский тип) не найден в коллекции.

IndexOf

Метод позволяет отыскать порядковый номер указанного дочернего типа объекта в коллекции.

Синтаксис:

```
Int32 IndexOf(Int32 objTypeID);
```

Параметры:

objTypeID

Идентификатор дочернего типа объекта.

Результат:

Порядковый номер указанного типа объекта или **-1**, если такой тип не найден в коллекции.

Load

Метод позволяет загрузить описание допустимого типа связи из указанного XML-документа.

Синтаксис:

```
void Load (XMLSettingsStorage storage, XmlNode node);
```

Параметры:

storage

Хранилище XML-документа, из которого требуется загрузить описание допустимого типа связи.

node

Узел XML, в котором хранятся свойства описания допустимого типа связи.

Save

Метод позволяет сохранить описание допустимого типа связи в указанный XML-документ.

Синтаксис:

```
void Save (XMLSettingsStorage storage, XmlNode node);
```

Параметры:

storage

Хранилище XML-документа, в которое требуется сохранить описание допустимого типа связи.

node

Узел XML, в котором требуется сохранить свойства описания допустимого типа связи.

SyncMetadata

Метод выполняет полную синхронизацию содержимого экземпляра класса с системным кэшем метаданных. При этом будут обновлён список дочерних типов объектов, устранены любые различия между свойствами правила и системными метаданными.

Синтаксис:

```
void Load (IUserSession session);
```

Параметры:

session

Сессия, в рамках которой выполняется работа с базой данных.

8.12.5 Класс **ChildObjectType**

8.12.5.1 *Пространство имён*

Intermech.Interfaces

В экземплярах данного класса хранятся идентификатор допустимого дочернего типа объектов, а также стартовое значение атрибута «Сортировка» (требуется для автоматической сортировки составов).

8.12.5.2 *Константы класса*

```
Int64 SortingValueDelta = 1000000
```

Приращение между стартовыми значениями атрибута "Сортировка" для разных дочерних типов объектов.

```
Int64 SortingValueSmallDelta = 1000
```

Приращение между стартовыми значениями атрибута "Сортировка" между объектами одного дочернего типа.

8.12.5.3 *Свойства класса*

```
Int32 ObjectTypeID { get; set; }
```

Идентификатор допустимого дочернего типа объектов.

```
Int64 StartSortingValue { get; set; }
```

Стартовое значение атрибута "Сортировка" для группы объектов состава указанного типа.

8.12.5.4 *Конструкторы*

Синтаксис:

```
ChildObjectType ();
```

Результат:

Создаётся пустой экземпляр класса.

Синтаксис:

```
ChildObjectType (Int32 objectTypeID, Int64 startSortingValue);
```

Параметры:

objectTypeID

Идентификатор допустимого дочернего типа объектов.

startSortingValue

Стартовое значение атрибута "Сортировка" для группы объектов состава указанного типа.

Результат:

Создаётся экземпляр класса, у которого заполняется все свойства.

8.12.5.5 *Методы класса*

Assign

Метод позволяет скопировать информацию из другого описания дочернего типа объекта в текущий.

Синтаксис:

```
void Assign(object source);
```

Параметры:

source

Объект, совместимый с типом **ChildObjectType**.

Clear

Метод очищает значение стартового значения атрибута «Сортировка».

Синтаксис:

```
void Clear();
```

Clone

Метод создаёт точную копию описания дочернего типа объекта.

Синтаксис:

```
object Clone ();
```

Результат:

Точная копия описания дочернего типа объекта.

Load

Метод позволяет загрузить описание дочернего типа объекта из указанного XML-документа.

Синтаксис:

```
void Load (XMLSettingsStorage storage, XmlNode node);
```

Параметры:

storage

Хранилище XML-документа, из которого требуется загрузить описание дочернего типа объекта.

node

Узел XML, в котором хранятся свойства описания дочернего типа объекта.

Save

Метод позволяет сохранить описание дочернего типа объекта в указанный XML-документ.

Синтаксис:

```
void Save (XMLSettingsStorage storage, XmlNode node);
```

Параметры:

storage

Хранилище XML-документа, в которое требуется сохранить описание дочернего типа объекта.

node

Узел XML, в котором требуется сохранить свойства описания дочернего типа объекта.

8.12.6 Перечислитель RulesEditorAccessRights

8.12.6.1 Пространство имён

Intermech.Interfaces

Данный перечислитель (набор битовых флажков) позволяет назначать права доступа в редакторе правил сортировки составов.

8.12.6.2 Значения перечислителя

ReadOnly = 0

У пользователя есть право просмотра коллекции правил.

CanSelectCurrentRule = 0x0001

Пользователь может назначать текущее правило сортировки в коллекции.

CanModifyChildObjects = 0x0010

Пользователь может выполнять изменение положения дочерних типов объектов в текущем правиле.

CanModifyCurrentRule = 0x0020

Пользователь может выполнять изменение текущего правила сортировки.

FullAccess = 0x0040 | RulesEditorAccessRights.CanModifyCurrentRule |

RulesEditorAccessRights.CanSelectCurrentRule

У пользователя есть полный доступ к правилам сортировки и отображения составов.

8.12.7 Интерфейс ICurrentUserAndRole

8.12.7.1 Пространство имён

Intermech.Interfaces.Client

Интерфейс, доступный на клиентском приложении, позволяет получать информацию об идентификаторе текущего пользователя, его роли, принадлежности его роли к администраторской роли, а также управлять текущим правилом сортировки и отображения составов. Получить данный интерфейс можно у статического класса **ServicesManager**.

8.12.7.2 Свойства интерфейса

```
Int64 UserID { get; }
```

Идентификатор текущего пользователя.

```
Int64 RoleID { get; }
```

Идентификатор роли текущего пользователя.

```
bool IsAdmin { get; }
```

Обладает ли текущий пользователь правами администратора.

```
CompositionsAutosortRule Rule { get; set; }
```

Текущее правило по сортировке и отображению составов.

Внимание! При получении данного правила возвращается его точная копия. Так что, если требуется внести изменения в само правило в run-time-режиме, то следует получить эту копию, выполнить в ней изменения, а затем присвоить эту копию данному свойству. При этом все окна клиентского приложения получают уведомление о том, что необходимо пересчитать составы из-за изменения текущего правила отображения и сортировки составов.

8.12.8 Получение информации о видимых типах связей

Для того, чтобы получить список идентификаторов (Int32, Guid) видимых типов связей для указанного типа объекта, следует получить у статического класса **ServicesManager** ссылку на интерфейс **ICurrentUserAndRole**, а затем вызвать метод **GetObjectTypeVisibleRelations** или **GetObjectTypeVisibleRelationsGuid** у свойства полученного интерфейса **ICurrentUserAndRole.Rule**.

8.12.9 Получение информации о порядке типов связей и объектов в составе

Для получения порядка типов связей, а также дочерних типов объектов для определённых типов связей также следует обратиться к свойству **Rule** интерфейса **ICurrentUserAndRole**.

8.13 Создание закладок в «Навигаторе»

Закладки в «Навигаторе» назначаются на определённые категории узлов или на категории узлов плюс типы (например, объектов). Рассмотрим по-порядку, каким образом можно создать и отобразить свою закладку на свою категорию узла.

8.13.1 Создание элемента управления с закладкой

Создаём элемент управления, унаследованный от **System.Windows.Forms.UserControl**. Добавляем в него реализацию интерфейса **Intermech.Navigator.Views.IView**. Например:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Text;
using System.Windows.Forms;
using Intermech.Navigator.Views;
using Intermech.Navigator.Interfaces;
using Intermech.Interfaces.Client;
using Intermech.Interfaces;

namespace Intermech.Navigator.Samples
{
    // Пример закладки для главного узла информационного пространства
    public partial class MainNodePage : UserControl, IView
    {
        // Сервис именованных значков
        private INamedImageList _namedImageList;
        // Конструктор
        public MainPage()
        {
            // Инициализировать компоненты
            InitializeComponent();
        }
        #region Реализация интерфейса IView
        // Инициализировать закладку
        // items - Коллекция выделенных элементов
        // provider - Контейнер сервисов
        public void Initialize(ISelectedItems items, IServiceProvider provider)
        {

```



```

        // Получаем ссылку на сервис
        _namedImageList = ServicesManager.GetService(typeof(INamedImageList)) as INamedImageList;
    }
    // Активировать закладку
    // previousView - Предыдущая закладка
    public void Activate(IView previousView)
    {
    }
    // Деактивировать закладку
    // nextView - Следующая закладка
    public void Deactivate(IView nextView)
    {
    }
    // Заголовок закладки
    public string Caption
    {
        get { return "Тестовая страничка"; }
    }
    // Индекс значка
    public int ImageIndex
    {
        get { return (_namedImageList != null) ? _namedImageList.ImageIndex("imgMainPageIcon") : -1; }
    }
    // Порядковый номер закладки
    public int OrderID
    {
        get { return 1; }
    }
    #endregion
}
}

```

Для закладки требуются её порядковый номер, название и индекс изображения в общем списке именованных значков в клиентском приложении.

8.13.2 Создание и регистрация провайдера для закладки

Для того, чтобы закладку можно было отобразить в «Навигаторе», для неё должен быть создан специальный класс-провайдер. Создаём класс, реализуем в нём интерфейс **Intermech.Navigator.Views.IViewsProvider** :

```

using System;
using Intermech.Navigator.Interfaces;
using Intermech.Navigator.Views;
namespace Intermech.Navigator.GlobalNode
{
    internal class ViewProvider: IViewsProvider
    {
        #region Реализация интерфейса IViewsProvider
        public ViewsInfo GetViews (ISelectedItems items, IServiceProvider services)
        {
            if (items.Count != 1) return ViewsInfo.Empty;
            ViewsInfo viewsInfo = new ViewsInfo();
            viewsInfo.Add("MainNodePage", new ViewInfo(TriggerPriority.Basic,
                typeof(MainNodePage)));
            return viewsInfo;
        }
        #endregion
    }
}

```

В данном коде создаётся класс-провайдер. В метод GetViews на вход поступают коллекция выделенных элементов (текущие узлы в дереве или гриде, т.п.) и контейнер сервисов для данных выделенных элементов.

Далее выполняется проверка – если выделен не один элемент, то закладки отображаться не будут.

Затем в список добавляется ссылка на вновь созданный тип закладки – **MainNodePage**, а также её уникальное в пределах всей системы внутреннее имя **MainNodePage**.

Если требуется дать пользователю возможность отключать видимость данной закладки, то следует выполнить следующую строку кода:

```
Intermech.Navigator.Views.AdjustableViewsHelper.RegisterView("Samples.MainNodePage",
    "Тестовая страничка",
    "Тестовая страничка для главного узла информационного пространства",
    "Intermech.Samples",
    " imgMainPageIcon ",
    true,
    1);
```

Закладка будет зарегистрирована с указанным именем, примечанием, изображением и порядковым номером. При этом пользователь или администратор для настроек роли сможет включать/отключать отображение данной закладки.

Для регистрации провайдера на определённую категорию узлов следует выполнить код:

```
Intermech.Navigator.Holder.Factory.AddViewsProvider (Navigator.Consts.CategoryGlobalNode,
new Intermech.Navigator.GlobalNode.ViewProvider());
```

где

Navigator.Consts.CategoryGlobalNode – категория корневого узла информационного пространства.

8.14 Работа со службой уведомлений

Когда в универсальном клиенте происходят некоторые события (создание, изменение, удаление объектов, связей, т.п.), то элементы управления, а также службы клиента должны получать об этом уведомления. Для этих целей используется служба уведомлений.

Служба уведомлений - это класс, реализующий интерфейс `INotificationService`. Ссылку на эту службу можно получить у глобального контейнера сервисов `ServicesManager`:

```
// Получаем ссылку на службу уведомлений
```

```
INotificationService _svc = ServicesManager.GetService(typeof(INotificationService)) as
INotificationService;
```

После этого можно воспользоваться её методами и событиями.

Рассмотрим подробно все интерфейсы и вспомогательные классы, а также механизм подписки на уведомления с целью их обработки.

8.14.1 Интерфейс `INotificationService`

`INotificationService` – это ядро службы уведомлений. Данная служба позволяет рассылать уведомления и получать их. Ссылка на неё доступна в глобальном контейнере сервисов `ServicesManager`.

8.14.1.1 Пространство имен

`Intermech.Interfaces.Client`

8.14.1.2 События интерфейса:

`Intermech.Interfaces.Client.NotificationEventHandler OnAfterEvent`

Событие генерируется после вызова указанного события.

`Intermech.Interfaces.Client.NotificationEventHandler OnBeforeEvent`

Событие генерируется перед вызовом указанного события.

8.14.1.3 Методы интерфейса:

`FireEvent`

Извещает всех подписчиков о произошедшем событии.

Синтаксис:

```
void FireEvent (object sender; Intermech.Interfaces.Client.NotificationEventArgs e)
```

Параметры:

sender

Объект, рассылающий событие обновления.

e

Данные для события обновления.

Subscribe

Осуществляет подписку на обработку любых событий обновления.

Синтаксис:

```
void Subscribe (Intermech.Interfaces.Client.NotificationEventHandler eventHandler)
void Subscribe (string eventName;
Intermech.Interfaces.Client.NotificationEventHandler eventHandler)
```

Параметры:

eventName

Имя события обновления.

eventHandler

Делегат обработчика события обновления.

Unsubscribe

Осуществляет отписку от обработки любых событий обновления.

Синтаксис:

```
void Unsubscribe (Intermech.Interfaces.Client.NotificationEventHandler eventHandler)
void Unsubscribe (string eventName; Intermech.Interfaces.Client.NotificationEventHandler eventHandler)
```

Параметры:

eventName

Имя события обновления.

eventHandler

Делегат обработчика события обновления.

8.14.2 Делегат NotificationEventHandler

Обработчик каждого уведомления является делегатом типа NotificationEventHandler.

8.14.2.1 Пространство имен

Intermech.Interfaces.Client

8.14.2.2 Сигнатура

Синтаксис:

```
void NotificationEventHandler (object sender; Intermech.Interfaces.Client.NotificationEventArgs e)
```

Параметры делегата:

sender

Отправитель уведомления.

e

Аргументы уведомления.

8.14.3 Класс NotificationEventNames

Все уведомления в системе должны иметь уникальные имена. В ядре клиентской части есть класс NotificationEventNames, константы которого хранят в себе имена системных уведомлений.

8.14.3.1 Пространство имён

Intermech.Interfaces.Client

Рекомендуется использовать константы этого класса для именования стандартных уведомлений, вместо указания их строкового эквивалента – на случай, если название уведомления будет изменено, а также для улучшения читабельности исходного кода программы.

8.14.3.2 Константы класса

```
const string AttributeCreated = "AttributeCreated"
```

Уведомление **AttributeCreated** - созданы новые атрибуты.

```
const string AttributeChanged = "AttributeChanged"
```

Уведомление **AttributeChanged** – атрибуты изменены.

```
const string AttributeRemoved = "AttributeRemoved"
```

Уведомление **AttributeRemoved** – атрибуты удалены.

```
const string Attribute40bjTypeEvent = "Attribute40bjTypeEvent"
```

Уведомление **Attribute4ObjTypeEvent** – добавлены, изменены или удалены атрибуты для типа объектов.

`const string Attribute4RelTypeEvent = "Attribute4RelTypeEvent"`

Уведомление **Attribute4RelTypeEvent** – добавлены, изменены или удалены атрибуты для типа связи.

`const string ObjectTypesCreated = "ObjectTypesCreated"`

Уведомление **ObjectTypesCreated** – типы объектов созданы.

`const string ObjectTypesChanged = "ObjectTypesChanged"`

Уведомление **ObjectTypesChanged** – типы объектов изменены.

`const string ObjectTypesRemoved = "ObjectTypesRemoved"`

Уведомление **ObjectTypesRemoved** – типы объектов удалены.

`const string ManagedObjectsCreated = "ManagedObjectsCreated"`

Уведомление **ManagedObjectsCreated** – объекты созданы, управляемое событие.

`const string ObjectsCreated = "ObjectsCreated"`

Уведомление **ObjectsCreated** – объекты созданы.

`const string ObjectsChanged = "ObjectsChanged"`

Уведомление **ObjectsChanged** – объекты изменены.

`const string ObjectsRemoved = "ObjectsRemoved"`

Уведомление **ObjectsRemoved** – объекты удалены.

`const string ObjectsCheckedOut = "ObjectsCheckedOut"`

Уведомление **ObjectsCheckedOut** – объекты взяты на изменение.

`const string ObjectsCheckedIn = "ObjectsCheckedIn"`

Уведомление **ObjectsCheckedIn** – у объектов завершены изменения.

`const string ObjectsChangesCancelled = "ObjectsChangesCancelled"`

Уведомление **ObjectsChangesCancelled** – у объектов отменены изменения.

`const string RelationsCreated = "RelationsCreated"`

Уведомление **RelationsCreated** – связи созданы.

`const string RelationsChanged = "RelationsChanged"`

Уведомление **RelationsChanged** – связи изменены.

`const string RelationsRemoved = "RelationsRemoved"`

Уведомление **RelationsRemoved** – связи удалены.

`const string ManagedRelationsCreated = "ManagedRelationsCreated"`

Уведомление **ManagedRelationsCreated** – связи созданы, управляемое событие.

`const string ManagedRelationsInsert = "ManagedRelationsInsert"`

Уведомление **ManagedRelationsInsert** – связи добавлены в определённую позицию, управляемое событие.

`const string SubstitutesChanged = "SubstitutesChanged"`

Уведомление **SubstitutesChanged** – изменены связи, участвующие в допустимых заменах.

`const string SortRelationsChanged = "SortRelationsChanged"`

Уведомление **SortRelationsChanged** – изменены связи, имеющие атрибут «Сортировка».

`const string FiltrationChanged = "FiltrationChanged"`

Уведомление **FiltrationChanged** – изменены настройки фильтрации составов.

`const string ObjectTypeAndRelationFiltrationChanged =
"ObjectTypeAndRelationFiltrationChanged"`

Уведомление **ObjectTypeAndRelationFiltrationChanged** – изменены настройки фильтрации типов объектов и связей.

`const string ApplicationClosing = "ApplicationClosing"`

Уведомление **ApplicationClosing** – приложение завершает свою работу.

`const string RecentObjectsChanged = "RecentObjectsChanged"`

Уведомление **RecentObjectsChanged** – изменился список недавних объектов.

Помимо уникального имени каждое уведомление несёт с собой аргументы – экземпляры классов, унаследованных от класса `NotificationEventArgs`.

8.14.4 Класс `NotificationEventArgs`

Аргументы событий должны быть наследниками от базового класса `NotificationEventArgs`.

8.14.4.1 Пространство имён

`Intermech.Interfaces.Client`

В составе базового класса с аргументами есть два свойства – имя события и флажок, определяющий, требуется ли службе уведомлений генерировать события «Перед основным событием» и «После основного события» или нет.

8.14.4.2 Свойства класса

```
string EventName { get; }
```

Имя уведомления.

```
bool FirePrePostEvents { get; set; }
```

Если установить данный флажок в true, то служба уведомлений во время вызова метода FireEvent сначала обращается к обработчикам события OnBeforeEvent, затем генерирует данное событие, а после того обращается к обработчикам события OnAfterEvent.

8.14.5 Использование стандартных уведомлений

Если есть необходимость использовать стандартные уведомления, необходимо воспользоваться следующей таблицей, чтобы определить, аргументы какого типа можно отправлять вместе с уведомлением. Все типы аргументов, указанные в таблице, располагаются в пространстве имён Intermech.Interfaces.Client.

Название уведомления	Тип аргументов уведомления
AttributeCreated, AttributeChanged, AttributeRemoved, Attribute4ObjTypeEvent, Attribute4RelTypeEvent	DBAttributesEventArgs
ObjectTypesCreated, ObjectTypesChanged, ObjectTypesRemoved	DBObjectTypesEventArgs
ObjectsCreated, ObjectsChanged, ObjectsCheckedIn, ObjectsChangesCancelled, ObjectsRemoved	DBObjectEventArgs
ManagedObjectsCreated	DBObjectManagedEventArgs
ObjectsCheckedOut	DBObjectCheckOutEventArgs
RelationsCreated, RelationsChanged, RelationsRemoved, SortedRelationsChanged	DBRelationsEventArgs
ManagedRelationsCreated, ManagedRelationsInsert	DBRelationsManagedEventArgs
SubstitutesChanged	DBSubstitutesEventArgs
FiltrationChanged, RecentObjectsChanged	NotificationEventArgs
ObjectsFiltrationChanged	DBObjectFiltrationEventArgs
ObjectTypeAndRelationFiltrationChanged	DBObjectTypeAndRelationFiltrationEventArgs
ApplicationClosing	ApplicationClosingEventArgs

8.14.6 Создание собственного уведомления

Если возникает необходимость создать своё собственное уведомление, следует выполнить следующие действия:

1. Определить уникальное в пределах всей системы имя нового уведомления,

2. Если требуется, создать класс, унаследованный от NotificationEventArgs или любого из его наследников,
3. Создать обработчик(и) данного уведомления .

Создание собственного уведомления рассмотрим на примере реализации уведомления ApplicationClosing, генерируемого главной формой клиентского приложения при попытке её закрытия.

1. Определяем уникальное имя – **ApplicationClosing**,
2. Создаём новый класс – ApplicationClosingEventArgs : NotificationEventArgs, в котором объявляем дополнительное свойство bool Cancel (если выставить его значение в **true**, закрытие приложения будет отменено), создаём новый конструктор для этого класса:
ApplicationClosingEventArgs(string eventName, bool cancel)
3. Реализуем обработчики данного события в главной форме приложения и в плагилах, которым требуется получать уведомления о попытках закрыть клиентское приложение.

В следующем разделе рассмотрим, как можно выполнять подписку на события – получать управление в код обработчиков событий при их возникновении.

8.14.7 Подписка на события службы уведомлений

Подписка на события происходит с помощью методов Subscribe службы INotificationService. Программист обязан создать метод, совместимый аргументами с делегатом типа NotificationEventHandler, и в методе Subscribe должен указать ссылку на этот метод, а также, при необходимости, имя события, на которое выполняется подписка:

```
// Получаем ссылку на службу уведомлений
public void AppClosingEventHandler(object sender, NotificationEventArgs e)
{
    if (e.EventName == NotificationEventNames. ApplicationClosing)
    {
        // Выполнить какие-либо действия
        ...
        // Если требуется – запретить завершение работы клиентского приложения
        e.Cancel = true;
    }
}
// Получаем ссылку на службу уведомлений
INotificationService _svc = ServicesManager.GetService(typeof(INotificationService)) as
INotificationService;
// Подписываемся на все уведомления
_svc.Subscribe(new NotificationEventHandler(AppClosingEventHandler));
или
// Подписываемся на конкретное уведомление
_svc.Subscribe(NotificationEventNames. ApplicationClosing,
    new NotificationEventHandler(AppClosingEventHandler));
```

Чтобы отписаться от уведомлений, следует воспользоваться методом Unsubscribe службы уведомлений.

8.15 Работа с менеджером команд

Менеджер команд позволяет связывать элементы управления панелей управления и главного меню с именованными командами, управляет элементами управления, которые могут обрабатывать именованные команды (отслеживает их активность, запрашивает статусы команд, отправляет команды на исполнение).

Механизм для работы с именованными командами функционирует следующим образом:

1. Создаются элементы управления (кнопки на панелях управления, элементы главного меню),
2. Каждому такому элементу назначается уникальное имя команды,

3. Такие элементы регистрируются в менеджере команд, при этом для каждой именованной команды создаётся её описание, позволяющее управлять статусом данной команды (разрешена, видима, отмечена, т.п.),
4. В менеджере команд регистрируются элементы управления-«цели» – обработчики команд, например, дерево «Навигатора», гриды, окна,
5. Менеджер команд отслеживает переключение фокуса ввода между «целями», активной «цели» при необходимости отправляются запросы на проверку статусов именованных команд, а также сами команды на исполнение,
6. При необходимости элементы управления могут реализовать определённый интерфейс для того, чтобы при их активации текущая «цель» не изменялась (плавающие панели, т.п.).

Рассмотрим подробно весь механизм именованных команд и работу с менеджером команд.

8.15.1 Интерфейс ICommandManager

Основой механизма является служба ICommandManager, которая реализована в сборке Intermech.Bars и доступна в глобальном контейнере сервисов ServicesManager.

8.15.1.1 Пространство имен

Intermech.Bars

8.15.1.2 Свойства интерфейса:

Intermech.Bars.ICommandTarget ActiveTarget { get; set; }

Активный обработчик именованных команд.

8.15.1.3 Методы интерфейса:

Add

Метод регистрирует в менеджере указанные элементы управления, которые представляют собой источники именованных команд (у каждого такого элемента управления есть свойство с уникальным названием команды).

Синтаксис:

```
Intermech.Bars.ICommandState Add (Intermech.Bars.ButtonItemBase controls)
Intermech.Bars.ICommandState Add (string commandName;
Intermech.Bars.ButtonItemBase controls)
```

Параметры:

commandName

Уникальное название команды.

controls

Элементы управления.

Возвращаемое значение:

Элемент управления.

AddTarget

Метод регистрирует в менеджере обработчик именованных команд – любой класс или элемент управления, реализующий интерфейс ICommandTarget.

Синтаксис:

```
void AddTarget (Intermech.Bars.ICommandTarget target)
```

Параметры:

target

Обработчик команд от менеджера команд.

Execute

Метод даёт активному обработчику команд запрос на выполнение указанной именованной команды.

Синтаксис:

```
bool Execute (Intermech.Bars.ICommandState commandState)
```

Параметры:

`commandState`

Элемент управления.

Возвращаемое значение:

true, если команда была успешно выполнена.

FindCommand

Отыскать элемент управления, реализующий указанную команду.

Синтаксис:

```
Intermech.Bars.ICommandState FindCommand (string commandName)
```

Параметры:

`commandName`

Искомая команда.

Возвращаемое значение:

Элемент управления, реализующий указанную команду.

QueryStatus

Метод выполняет проверку состояния всех зарегистрированных (либо только указанных) именованных команд в активном обработчике команд. Для каждой команды запрашивается её статус – можно ли показывать, выполнять эту команду в активном обработчике. Результаты проверки сохраняются в описаниях команд.

Синтаксис:

```
void QueryStatus ()
```

```
void QueryStatus (Intermech.Bars.ICommandState commandState)
```

Параметры:

`commandState`

Элемент управления.

RemoveTarget

Метод удаляет из менеджера указанный обработчик именованных команд.

Синтаксис:

```
void RemoveTarget (Intermech.Bars.ICommandTarget target)
```

Параметры:

`target`

Удаляемый обработчик команд.

Таким образом, программист может реализовать обработчик команд, зарегистрировать его в менеджере команд и начать получать запросы статуса и выполнения команд. В дальнейшем можно удалить обработчик из менеджера команд.

Рассмотрим подробнее описание именованной команды – интерфейс `ICommandState`.

8.15.2 Интерфейс `ICommandState`

Элемент управления, поддерживающий работу с менеджером команд.

8.15.2.1 *Пространство имен*

`Intermech.Bars`

8.15.2.2 *Свойства интерфейса:*

```
bool Checked { get; set; }
```

Элемент управления отмечен или нет.

```
string CommandName { get; }
```

Уникальное в пределах всей системы название команды.

```
bool Enabled { get; set; }
```

Элемент управления разрешён или запрещён.

```
Int32 ImageIndex { get; set; }
```

Индекс изображения для элемента управления.

```
object Tag { get; set; }
```

Дополнительное значение для элемента управления.

```
string Text { get; set; }
```


Заголовок элемента управления.

```
string ToolTipText { get; set; }
```

Текст подсказки для элемента управления.

```
bool Visible { get; set; }
```

Элемент управления видим или нет.

Меняя свойства в описании команды, программист контролирует поведение элемента управления, связанного с данной командой, например, разрешает или запрещает кнопки на панели управления, меняет им текст подсказок, управляет их видимостью.

Далее будут рассмотрены интерфейсы ICommandTarget и ICommandTarget2 – обработчики именованных команд.

8.15.3 Интерфейс ICommandTarget

Когда от какого-либо элемента управления (дерево, грид, т.п.) требуется поддержка именованных команд (управление их статусом, выполнение этих команд), то данный элемент должен реализовать интерфейс ICommandTarget, а также зарегистрироваться в менеджере команд.

8.15.3.1 Пространство имен

Intermech.Bars

8.15.3.2 Методы интерфейса:

Execute

Выполнить команду.

Синтаксис:

```
bool Execute (Intermech.Bars.ICommandState commandState)
```

Параметры:

commandState

Команда.

Возвращаемое значение:

true, если команда выполнена успешно.

QueryStatus

Установить статус команде.

Синтаксис:

```
bool QueryStatus (Intermech.Bars.ICommandState commandState)
```

Параметры:

commandState

Команда.

Возвращаемое значение:

true, статус команды установлен.

8.15.4 Интерфейс ICommandTarget2

Расширение интерфейса команд ICommandTarget. Поддерживает методы начала и конца цикла опроса.

8.15.4.1 Пространство имен

Intermech.Bars

8.15.4.2 Реализуемые интерфейсы:

Intermech.Bars.ICommandTarget

8.15.4.3 Методы интерфейса:

BeginQuery

Начинается опрос состояний команд.

Синтаксис:

```
void BeginQuery ()
```

EndQuery

Опрос состояний команд закончен.

Синтаксис:

void EndQuery ()

Остальные методы унаследованы от базового интерфейса ICommandTarget.

8.16 Панели инструментов универсального клиента

У каждой панели инструментов есть свой уникальный Guid идентификатор, по которому панели можно отыскать, сохранить, а затем восстановить её состояние.

Доступ к панелям управления осуществляется с помощью экземпляра класса BarManager, доступного в глобальном контейнере сервисов ServicesManager. Рассмотрим подробнее функционал данного класса.

8.16.1 Класс BarManager

Доступ к панелям управления осуществляется с помощью экземпляра класса BarManager, доступного в глобальном контейнере сервисов ServicesManager.

8.16.1.1 Пространство имен

Intermech.Bars

8.16.1.2 Реализуемые интерфейсы:

System.Collections.IComparer

8.16.1.3 Свойства класса:

Intermech.Bars.ToolBarContainer BottomContainer { get; set; }

Нижний контейнер, в котором можно размещать панели инструментов.

bool EnableContextMenu { get; set; }

Если установить значение в true, у пользователя будет возможность вызывать контекстное меню для настройки панели инструментов – прятать или добавлять элементы управления, размещённые на панели.

bool FormHasFocus { get; }

true, если форма, которой принадлежат панели инструментов, является активной формой.

Intermech.Bars.ToolBarContainer LeftContainer { get; set; }

Левый контейнер, в котором можно размещать панели инструментов.

Intermech.Bars.MenuBar MenuBar { get; }

Главное меню приложения.

System.Windows.Forms.Form OwnerForm { get; set; }

Форма-владелец панелей инструментов.

Intermech.Bars.IToolBarRenderer Renderer { get; set; }

Интерфейс, который отвечает за прорисовку панелей инструментов (тема оформления панелей).

Intermech.Bars.ToolBarContainer RightContainer { get; set; }

Правый контейнер, в котором можно размещать панели инструментов.

Intermech.Bars.ToolBarContainer TopContainer { get; set; }

Верхний контейнер, в котором можно размещать панели инструментов.

8.16.1.4 События класса:

System.EventHandler RendererChanged

Событие генерируется при изменении темы оформления панелей инструментов.

8.16.1.5 Методы класса:

AddToolbar

Метод добавляет панель инструментов в менеджер, если панель ещё не была добавлена.

Синтаксис:

void AddToolbar (Intermech.Bars.ToolBar toolbar)

void AddToolbar (Intermech.Bars.ToolBar toolbar;

System.Windows.Forms.DockStyle dockStyle)

Параметры:

toolbar

Добавляемая панель инструментов.

dockStyle

Каким образом панель будет размещаться в контейнере.

FindContainer

Метод ищет в менеджере контейнер панелей инструментов с указанным идентификатором.

Синтаксис:

```
Intermech.Bars.ToolBarContainer FindContainer (System.Guid guid)
```

Параметры:

guid

Идентификатор контейнера панелей инструментов.

Возвращаемое значение:

Контейнер панелей инструментов или null.

FindSuitableContainer

Метод ищет в менеджере наиболее подходящий контейнер панелей инструментов с указанным стилем размещения.

Синтаксис:

```
Intermech.Bars.ToolBarContainer FindSuitableContainer (System.Windows.Forms.DockStyle dockStyle)
```

Параметры:

dockStyle

Искомый стиль размещения.

Возвращаемое значение:

Подходящий контейнер панелей инструментов или null.

FindToolbar

Метод ищет в менеджере панель инструментов с указанным идентификатором или именем.

Синтаксис:

```
Intermech.Bars.Toolbar FindToolbar (System.Guid guid)
```

```
Intermech.Bars.Toolbar FindToolbar (string name)
```

Параметры:

guid

Уникальный идентификатор искомой панели инструментов.

name

Название искомой панели инструментов.

Возвращаемое значение:

Найденная панель инструментов или null.

GetContainers

Метод возвращает массив всех контейнеров панелей инструментов.

Синтаксис:

```
Intermech.Bars.ToolBarContainer[] GetContainers ()
```

Возвращаемое значение:

Массив всех контейнеров панелей инструментов, доступных в менеджере.

GetLayout

Метод возвращает строку, в которой сохранено текущее состояние всех панелей инструментов.

Синтаксис:

```
string GetLayout ()
```

```
string GetLayout (bool includeItemVisibility)
```

Параметры:

includeItemVisibility

Включать ли в строку с состоянием информацию о видимости элементов управления панелей.

Возвращаемое значение:

Строка, в которой сохранено текущее состояние всех панелей инструментов.

GetToolBars

Метод возвращает массив всех панелей инструментов.

Синтаксис:

```
Intermech.Bars.Toolbar[] GetToolBars ()
```

Возвращаемое значение:

Массив всех панелей инструментов, доступных в менеджере.

GetToolbarsList

Получить список доступных панелей управления, которые можно скрывать/показывать.

Синтаксис:

```
System.Collections.Generic.List<ToolBar> GetToolbarsList ()
```

Возвращаемое значение:

Список доступных панелей управления, которые можно скрывать/показывать.

RemoveToolBar

Метод позволяет удалить панель из менеджера.

Синтаксис:

```
void RemoveToolBar (Intermech.Bars.ToolBar toolbar)
```

Параметры:

toolbar

Удаляемая из менеджера панель инструментов.

SetLayout

Метод позволяет восстановить состояние панелей инструментов из строки (в которую ранее было сохранено это состояние).

Синтаксис:

```
void SetLayout (string layout)
```

Параметры:

layout

Сохранённое в виде строки состояние всех панелей инструментов.

8.16.1.6 Статические методы класса:

Merge

Статический метод класса позволяет объединить панель target инструментов с панелью source, результат – одна панель target.

Синтаксис:

```
static void Merge (Intermech.Bars.ToolBar source; Intermech.Bars.ToolBar target)
```

Параметры:

source

Исходная панель инструментов.

target

Панель инструментов, которая будет результатом объединения.

UndoMerge

Статический метод класса позволяет разъединить составную панель (созданную методом Merge).

Синтаксис:

```
static void UndoMerge (Intermech.Bars.ToolBar toolbar)
```

Параметры:

toolbar

Составная панель инструментов, которую требуется разъединить.

Таким образом, получив ссылку на BarManager, программист может управлять всеми доступными панелями инструментов – добавлять, изменять, получать список, удалять панели, сохранять и восстанавливать состояние всех панелей.

8.16.2 Класс ToolBarContainer

В менеджере панелей инструментов есть четыре контейнера, расположенных сверху, слева, справа и внизу формы, которой принадлежит менеджер. Панели инструментов можно размещать в любом из контейнеров, либо делать их «плавающими» - не принадлежащими ни одному из контейнеров.

8.16.2.1 Пространство имен

Intermech.Bars

8.16.2.2 Свойства класса:

```
System.Drawing.Color BackColor { get; set; }
```

Цвет фона контейнера панелей инструментов.

`System.Drawing.Image BackgroundImage { get; set; }`

Фоновое изображение для контейнера.

`System.Drawing.Color ForeColor { get; set; }`

Цвет текста для контейнера.

`System.Guid Guid { get; set; }`

Уникальный идентификатор контейнера.

`Intermech.Bars.BarManager Manager { get; set; }`

Менеджер панелей инструментов, которому принадлежит данный контейнер панелей.

`string Text { get; set; }`

Название контейнера панелей.

8.16.2.3 Методы класса:

GetNextFreeDockLine

Метод отыскивает номер свободной строки для размещения панелей инструментов.

Синтаксис:

`Int32 GetNextFreeDockLine ()`

Возвращаемое значение:

Номер первой свободной строки в контейнере.

8.16.3 Класс ToolBar

Класс представляет собой панель инструментов. Каждой панели инструментов можно назначить контейнер, в котором она будет размещена.

8.16.3.1 Пространство имен

Intermech.Bars

8.16.3.2 Реализуемые интерфейсы:

`Intermech.Bars.IButtonsSite`

`Intermech.Bars.IPopupMenuHost`

8.16.3.3 Свойства класса:

`Intermech.Bars.TopLevelMenuItemBase ActionsButton { get; set; }`

Список кнопок, размещённых на панели инструментов.

`bool AddRemoveButtonsVisible { get; set; }`

Значение true позволяет отображать команду "Добавить/удалить кнопки" в контекстном меню панели инструментов (тем самым позволяя пользователю выполнять настройку панели, скрывая или отображая её элементы управления).

`bool AllowHorizontalDock { get; set; }`

Значение true разрешает панели инструментов размещаться в контейнере горизонтально.

`bool AllowMerge { get; set; }`

Значение true разрешает панели инструментов объединяться с другими панелями.

`bool AllowVerticalDock { get; set; }`

Значение true разрешает панели инструментов размещаться в контейнере вертикально.

`System.Drawing.Color BackColor { get; set; }`

Цвет фона панели инструментов.

`System.Drawing.Image BackgroundImage { get; set; }`

Фоновое изображение для панели инструментов.

`bool Closable { get; set; }`

Значение true разрешает панели инструментов отображать кнопку **Заккрыть**, когда панель находится в плавающем состоянии (не размещена ни в одном контейнере).

`Int32 DockLine { get; set; }`

Позволяет управлять номером строки в контейнере, в которой размещается данная панель инструментов.

`Int32 DockOffset { get; set; }`

Смещение в пикселах относительно начала строки контейнера, по которому размещается панель в данной строке.

```
bool DrawActionButton { get; set; }
```

Значение true позволяет панели инструментов отображать кнопку для вызова контекстного меню по настройке панели.

```
Intermech.Bars.ToolBarLayout Flow { get; set; }
```

Данный перечислитель позволяет указать, как именно располагаются элементы управления в панели – горизонтально или вертикально.

```
System.Drawing.Color ForeColor { get; set; }
```

Цвет текста для панели инструментов.

```
bool FullMenus { get; set; }
```

Значение true уведомляет о том, что панель отображает свои меню полностью, не скрывая редко используемые его пункты.

```
System.Guid Guid { get; set; }
```

Уникальный идентификатор панели инструментов.

```
bool Hidden { get; set; }
```

Значение true скрывает панель, однако сам элемент управления панели считается видимым.

```
Intermech.Bars.ToolBarItemBase HighlightedItem { get; set; }
```

Текущий выделенный элемент управления на панели.

```
System.Windows.Forms.ImageList ImageList { get; set; }
```

Список изображений, который будет использоваться для элементов управления, принадлежащих панели.

```
bool IsFloating { get; }
```

Значение true уведомляет о том, что панель находится в «плавающем» состоянии (не прикреплена ни к какому контейнеру панелей).

```
bool IsOpen { get; }
```

Значение true уведомляет о том, что панель не находится в «плавающем» состоянии и является видимой, либо не является скрытой.

```
Intermech.Bars.ToolBarItemBaseCollection Items { get; }
```

Список всех элементов управления, размещённых на панели инструментов.

```
Intermech.Bars.ToolBarContainer LastFixedContainer { get; }
```

Ссылка на последний контейнер, которому принадлежала данная панель.

```
System.Drawing.Size MaximumFloatingSize { get; set; }
```

Максимально допустимый размер панели в «плавающем» состоянии.

```
Intermech.Bars.ToolBar MergedToolBar { get; }
```

Ссылка на панель управления, с которой данная панель была объединена, или null.

```
System.Drawing.Size MinimumFloatingSize { get; set; }
```

Минимально допустимый размер панели в «плавающем» состоянии.

```
bool Movable { get; set; }
```

Значение true позволяет пользователю перемещать панель управления.

```
Intermech.Bars.ToolBarOverflow Overflow { get; set; }
```

Значения данного перечислителя задают, каким образом панель будет отображать элементы управления, если они не могут быть полностью размещены на панели: – `ToolBarOverflow.Chevron` – не уместящиеся элементы скрываются, а панель отображает специальную кнопку в конце для выбора этих скрытых элементов (взамен каких-либо отображающихся), – `ToolBarOverflow.Hide` – не уместящиеся элементы не отображаются на панели, – `ToolBarOverflow.Wrap` – панель меняет свою ширину и высоту таким образом, чтобы все её элементы управления могли полностью отобразиться на панели.

```
Intermech.Bars.IToolBarRenderer Renderer { get; set; }
```

Данный интерфейс отвечает за прорисовку панели инструментов в состоянии, когда панель пристыкована к контейнеру панелей.

```
bool Resizable { get; set; }
```

Значение true уведомляет о том, что панель позволяет пользователю изменять её размеры с помощью мыши.

```
bool ShowShortcutsInToolTips { get; set; }
```

Значение true позволяет панели отображать во всплывающих подсказках дополнительную информацию о «горячих» клавишах, назначенных её элементам управления.

```
Intermech.Bars.ToolBarSituation Situation { get; }
```

Текущее состояние панели инструментов: - `ToolBarSituation.Contained` – панель находится в составе другой панели, - `ToolBarSituation.Floating` – панель находится в «плавающем» состоянии, - `ToolBarSituation.Standalone` – панель пристыкована к контейнеру панелей.

```
bool Stretch { get; set; }
```

Значение true указывает панели на то, что она должна занимать всю строку в контейнере панелей (или всё оставшееся свободное в этой строке место).

```
Intermech.Bars.ToolBarItemBase StretchItem { get; set; }
```

Ссылка на элемент управления, который занимает всё свободное место внутри панели инструментов.

```
bool Tearable { get; set; }
```

Значение true позволяет пользователю переводить панель в «плавающее» состояние и обратно.

```
string Text { get; set; }
```

Название панели инструментов.

```
Intermech.Bars.ToolBarTextAlign TextAlign { get; set; }
```

Способ размещения текста у элементов управления относительно изображений у этих элементов: - `ToolBarTextAlign.Side` – текст размещается справа от изображений, - `ToolBarTextAlign.Underneath` – текст размещается под изображениями.

```
bool Vertical { get; }
```

Указывает, является ли расположение панели инструментов вертикальным.

```
Intermech.Bars.IToolBarRenderer WorkingRenderer { get; }
```

Ссылка на интерфейс, который отвечает за текущую прорисовку панели инструментов (зависит от того, в каком состоянии находится панель – в пристыкованном или в «плавающем»).

8.16.3.4 Методы класса:

Float

Метод переводит панель в «плавающий» режим в указанной позиции.

Синтаксис:

```
void Float (Intermech.Bars.BarManager manager; System.Drawing.Point desktopLocation)
```

Параметры:

`manager`

Менеджер панелей инструментов, который управляет данной панелью.

`desktopLocation`

Позиция (в координатах рабочего стола), на которой разместится панель инструментов.

GetItemAt

Метод отыскивает элемент управления, принадлежащий панели инструментов, и расположенный по указанным координатам.

Синтаксис:

```
Intermech.Bars.ToolBarItemBase GetItemAt (System.Drawing.Point position)
```

Параметры:

`position`

Координаты искомого элемента управления.

Возвращаемое значение:

Ссылка на найденный элемент управления или null.

Redock

Метод выполняет повторную стыковку панели с указанным элементом управления.

Синтаксис:

```
void Redock (System.Windows.Forms.Control container)
```

Параметры:

`container`

Элемент управления, к которому пристыковывается панель инструментов.

8.16.4 Создание новых панелей инструментов

Для того, чтобы создать новую панель инструментов, программисту требуется выполнить следующие действия:

1. Получить ссылку на общий экземпляр класса `BarManager` у глобального контейнера сервисов `ServicesManager`;
2. Создать новый экземпляр класса `ToolBar`, добавить в него элементы управления (кнопки, выпадающие меню, комбо-боксы);

3. Добавить вновь созданную панель в список панелей методом `BarManager.AddToolbar`;
4. При необходимости перевести панель в «плавающий режим».

8.16.5 Поиск существующих панелей инструментов

У каждой панели управления есть уникальный Guid идентификатор, по которому можно отыскать эту панель с помощью метода `BarManager.FindToolbar`.

Более подробно с панелями инструментов можно ознакомиться по примеру, поставляемому с данной документацией.

8.17 Создание окон, поддерживающих докинг и сохранение состояния

Если требуется разработать окно, которое будет способно работать с системой докинга универсального клиента, программист должен выполнить следующие подготовительные действия:

1. Создать элемент управления, унаследованный от `UserControl`;
2. Открыть код полученного класса и заменить строку :
`UserControl` на строку : `Intermech.Docking.DockControl` (заменить базовый класс);
3. Перекомпилировать код.

В результате получена заготовка элемента управления, которую далее следует доработать для полноценной работы в универсальном клиенте.

У каждого окна, поддерживающего докинг, есть свойства, которые следует переопределить. Следует обратить внимание на то, что окна докинга связаны с именованным списком изображений. Т.е., если для окна требуется отобразить изображение, следует указать индекс этого изображения из списка именованных значков.

8.17.1.1 Свойства класса `DockControl`

```
Guid Guid { get; set; }
```

Уникальный идентификатор окна. По этому идентификатору идёт сохранение и восстановление состояния окна, а также поиск окна в списке.

```
string Text { get; set; }
```

Заголовок окна.

```
bool ShowImageInDocumentTab { get; set; }
```

Если установить значение в `true`, то окно может отображать значок рядом со своим заголовком.

```
Int32 TabImageIndex { get; set; }
```

Индекс отображаемого рядом с заголовком окна изображения из именованного списка изображений. Значение `-1` означает отсутствие изображения.

Рекомендуется задавать значения указанных свойств в конструкторе нового элемента управления.

8.17.2 Добавление возможности сохранять и восстанавливать состояние

Для того, чтобы форма могла сохранять своё состояние, она должна перекрыть метод `GetPersistString` для получения своего текущего состояния, а также подписаться в системной службе `IContentProvider` на обработку события `ContentCallback` для восстановления состояния.

Подробности следует изучить в коде соответствующего примера, поставляемого с данной документацией.

8.18 Работа с диспетчером событий

Чтобы увязать в единое целое систему сообщений от клавиатуры, мышки, `drag'n'drop` от любых элементов управления, используется служба диспетчера событий и события.

Суть механизма заключается в следующем:

1. Элемент управления, который генерирует события, реализует интерфейс `IIOSource`;
2. Этот элемент получает ссылку на диспетчер событий `IIODispatcher` и регистрируется в нём как источник событий;
3. Элементы управления, которые заинтересованы в приёме событий, реализуют интерфейс `IIODestination`;

4. Эти элементы управления получают ссылку на `IODispatcher` и регистрируются в нём как приёмники событий;
5. Источник событий генерирует события типа `IOEvent`;
6. Диспетчер событий отправляет события всем подписчикам до тех пор, пока событие не будет обработано; если событие широковещательное, то оно рассылается всем подписчикам.

С помощью этого механизма можно рассылать типы событий, описанные в перечислителе `IOEventType`. Рассмотрим подробно всё, что связано с диспетчером событий.

8.18.1 Перечислитель `IOEventFlags`

Данный перечислитель характеризует рассылаемое событие. Перечислитель является набором битовых флажков – [Flags].

8.18.1.1 Пространство имён

`Intermech.Navigator.Interfaces`

8.18.1.2 Элементы перечислителя

`efNone` = 0x00000000
 Никаких признаков у события нет.
`efProcessed` = 0x00000001
 Событие обработано.
`efBroadcast` = 0x00000002
 Событие является широковещательным.

8.18.2 Перечислитель `IOEventType`

Данный перечислитель описывает тип рассылаемого события.

8.18.2.1 Пространство имён

`Intermech.Navigator.Interfaces`

8.18.2.2 Элементы перечислителя

`evNone` = 0x00000000
 Нет события.
`evKeyDown` = 0x00000001
 Нажата клавиша.
`evKeyUp` = 0x00000002
 Отпущена клавиша.
`evMouseClicked` = 0x00010000
 Нажата кнопка мыши.
`evMouseDoubleClick` = 0x00020000
 Двойной клик мышью.
`evDragDropEnter` = 0x01000000
 Событие drag'n'drop пришло в клиентскую область элемента управления.
`evDragDropOver` = 0x02000000
 Событие drag'n'drop - выполняется перетаскивание в клиентской области элемента управления.
`evDragDropLeave` = 0x04000000
 Событие drag'n'drop - перетаскивание ушло из клиентской области элемента управления.
`evDragDrop` = 0x08000000
 Событие drag'n'drop - перетаскивание завершилось в клиентской области элемента управления.

8.18.3 Перечислитель `IOEventTypes`

Данный перечислитель описывает типы событий, которые поддерживаются элементом управления – обработчиком (`IODestination`). Перечислитель является набором битовых флажков - [Flags].

8.18.3.1 Пространство имён

`Intermech.Navigator.Interfaces`

8.18.3.2 Элементы перечислителя

evNone = 0x00000000

Нет события.

evKeyDown = 0x00000001

Нажата клавиша.

evKeyUp = 0x00000002

Отпущена клавиша.

evMouseClicked = 0x00010000

Нажата кнопка мыши.

evMouseDoubleClick = 0x00020000

Двойной клик мышью.

evDragDropEnter = 0x01000000

Событие drag'n'drop пришло в клиентскую область элемента управления.

evDragDropOver = 0x02000000

Событие drag'n'drop - выполняется перетаскивание в клиентской области элемента управления.

evDragDropLeave = 0x04000000

Событие drag'n'drop - перетаскивание ушло из клиентской области элемента управления.

evDragDrop = 0x08000000

Событие drag'n'drop - перетаскивание завершилось в клиентской области элемента управления.

8.18.4 Интерфейс IOEvent

Интерфейс описывает событие, произошедшее в универсальном клиенте.

8.18.4.1 Пространство имён

Intermech.Navigator.Interfaces

У каждого события есть тип события, данные, а также ссылка на источник события.

8.18.4.2 Свойства интерфейса

IOSource Source { get; }

Ссылка на интерфейс, который является источником данного события.

IOEventFlags EventFlags { get; set; }

Флажки данного события.

IOEventType EventType { get; set; }

Тип события.

object eventData { get; }

Данные события. Трактуются в зависимости от значения Event:

evKeyDown, evKeyUp:

в поле хранится объект типа KeyEventArgs,

evMouseClicked, evMouseDoubleClick:

в поле хранится объект типа MouseEventArgs или EventArgs,

evDragDrop*:

в поле хранится объект типа DragEventArgs.

object Tag { get; set; }

Пользовательские данные, которые можно прикрепить к событию.

8.18.5 Класс IOEvent

Когда требуется сгенерировать событие, следует создавать экземпляры данного класса.

8.18.5.1 Пространство имён

Intermech.Client.Core

Данный класс реализует интерфейс Intermech.Navigator.Interfaces.IOEvent.

Свойства данного класса идентичны свойствам интерфейса IOEvent.

8.18.6 Интерфейс IODispatcher

Диспетчер событий позволяет выполнять следующие функции:

1. Регистрирует обработчики событий;
2. Удаляет информацию об обработчиках событий;
3. Позволяет передавать события от источников событий к их обработчикам (выполнять события).

8.18.6.1 Пространство имён

Intermech.Navigator.Interfaces

Интерфейс диспетчера событий.

8.18.6.2 Методы интерфейса

RegisterDestination

Метод выполняет регистрацию указанного обработчика событий в диспетчере событий.

Синтаксис:

```
void RegisterDestination (IIODestination Destination);
```

Параметры:

Destination

Интерфейс, реализующий обработчик событий типа IIOEvent.

UnregisterDestination

Метод выполняет удаление указанного обработчика событий из диспетчера событий.

Синтаксис:

```
void UnregisterDestination (IIODestination Destination);
```

Параметры:

Destination

Удаляемый из диспетчера событий интерфейс, реализующий обработчик событий типа IIOEvent.

ProcessEvent

Метод выполняет обработку указанного события в зарегистрированных обработчиках событий.

Синтаксис:

```
void ProcessEvent (IIOEvent Event);
```

Параметры:

Event

Обрабатываемое событие.

8.18.7 Класс IODispatcher

Когда элементу управления требуется свой персональный диспетчер, например, в окне есть свои элементы управления, которые генерируют события типа IIOEvent, и само окно должно их обрабатывать, оно создаёт у себя экземпляр данного класса, реализует интерфейс IIODestination и регистрируется в указанном диспетчере событий. При этом диспетчер событий должен быть добавлен в контейнер сервисов, который данное окно передаёт своим дочерним элементам управления (чтобы они могли отправлять свои события данному диспетчеру).

8.18.7.1 Пространство имён

Intermech.Client.Core.IODispatcher

Данный класс реализует интерфейс Intermech.Navigator.Interfaces.IODispatcher.

Методы данного класса идентичны методам интерфейса IODispatcher.

8.18.8 Интерфейс IIOSource

Элемент управления, который генерирует события типа IIOSource, должен реализовать данный интерфейс.

8.18.8.1 Пространство имён

Intermech.Navigator.Interfaces

Интерфейс источника событий.

8.18.8.2 Свойства интерфейса

```
object Control { get; set; }
```

Элемент управления, который является источником событий типа IIOSource.

```
IServiceProvider Services { get; set; }
```

Контейнер сервисов, который передаётся вместе с данными события. Это контекст для передаваемых данных.

```
ISelectedItems SelectedItems { get; set; }
```

Коллекция выделенных элементов пространства навигации, с которыми произошло событие.

8.18.9 Интерфейс IIODestination

Элемент управления, который обрабатывает события типа IIOSource, должен реализовать данный интерфейс.

8.18.9.1 Пространство имён

Intermech.Navigator.Interfaces

Интерфейс источника событий.

8.18.9.2 Свойства интерфейса

IOEventTypes SupportedEvents { get; set; }

Битовый перечислитель, характеризующий список событий, которые могут быть получены данным обработчиком. События, тип которых не указан в данном свойстве, в данный обработчик поступать от диспетчера событий не будут.

8.18.9.3 Методы интерфейса

ProcessEvent

Метод вызывается для обработки указанного события.

Синтаксис:

```
bool ProcessEvent (IIOEvent Event);
```

Параметры:

Event

Обрабатываемое событие.

Результат:

Значение **true** означает то, что событие было обработано.

Более подробно работу с диспетчером событий можно найти в примере, поставляемом с данной документацией.

8.19 Создание новых типов узлов для «Навигатора»

Узел **Навигатора** – это элемент пространства навигации, который содержит какие-либо данные, привязан к источнику данных, умеет обновлять своё содержимое, создавать свои дочерние элементы пространства навигации. У каждого узла есть две коллекции колонок – колонки по умолчанию и допустимые колонки.

Пространство навигации содержит корневой узел, для которого есть описание – класс, реализующий интерфейс IDescriptor, унаследованный от INodeItems. Дескриптор корневого узла умеет выполнять ряд действий, а именно:

1. Создаёт класс, хранящий идентификатор корневого узла (INodeID);
2. Создаёт дочерние элементы пространства навигации (классы, реализующие INode);
3. Возвращает данные запрошенного типа по идентификатору узла INodeID;
4. **Отражает** описания колонок NodeColumn на идентификаторы атрибутов из источника данных;
5. Возвращает массив значений для идентификатора INodeID;
6. Возвращает анализаторы для определённых событий **Навигатора**;
7. Возвращает ссылки на определённые типы служб;
8. Сохраняет своё состояние в контейнеры значений и умеет восстанавливать состояние из них.

Корневой узел может содержать дочерние элементы пространства навигации (каждый из которых тоже может иметь свою иерархию дочерних элементов).

У каждого элемента пространства навигации (узла) есть класс, в котором хранятся свойства этого элемента, например информация, полученная из источника данных, и т.п. Данные классы реализуют интерфейс INodeID.

Элементы навигации пространства представляют собой классы, реализующие интерфейс `INode` (и ряд вспомогательных интерфейсов). Дочерние элементы создаются их родительскими элементами с помощью метода `GetChild` на основании идентификаторов дочерних элементов (`INodeID`).

Рассмотрим подробно все классы, интерфейсы и перечислители, которые используются в пространстве навигации универсального клиента.

8.19.1 Класс `NodeColumn`

Данный класс представляет собой колонку **Навигатора**. Основное свойство – идентификатор атрибута, получаемого из источника данных, например, атрибута объекта или связи.

8.19.1.1 Пространство имен

`Intermech.Navigator.Interfaces`

8.19.1.2 Реализуемые интерфейсы:

`Intermech.Interfaces.IAssignable`

`System.ICloneable`

`System.IComparable`

`System.IComparable<NodeColumn>`

8.19.1.3 Свойства класса:

`Intermech.FieldTypes AttrType { get; }`

Тип данных атрибута, связанного с колонкой.

`string Caption { get; }`

Возвращает заголовок колонки.

`Intermech.Kernel.Search.ColumnContents Contents { get; set; }`

Содержимое ячеек колонки, которое будет запрашиваться у источника данных.

`System.Type DataType { get; }`

Тип данных, содержащихся в ячейках колонки.

`Int32 GroupIndex { get; set; }`

Порядковый номер в группе колонок или -1, если колонка не участвует в группировке колонок.

`string Hint { get; }`

Возвращает подсказку для заголовка колонки.

`object ID { get; }`

Идентификатор колонки. Например, может быть равен идентификатору или названию атрибута источника данных.

`string Key { get; }`

Уникальный строковый ключ колонки для её быстрого поиска в коллекции колонок. Ключ состоит из идентификатора схемы, которой принадлежит колонка, и идентификатора самой колонки.

`System.Guid SchemeGuid { get; }`

Идентификатор схемы, которой принадлежит колонка.

`string ShortCaption { get; }`

Возвращает краткий заголовок колонки.

`Int32 SortIndex { get; set; }`

Порядковый номер колонки в списке сортируемых колонок или -1, если колонка не участвует в сортировке.

`Intermech.Navigator.Interfaces.NodeColumnSortOrder SortOrder { get; set; }`

Возвращает или устанавливает направление сортировки данных в колонке.

`bool SystemAttr { get; set; }`

Если true, то атрибут, который описан в колонке, является системным.

`object Tag { get; set; }`

Дополнительные данные для колонки.

`Intermech.Navigator.Interfaces.CellTransformationMode TransformationMode { get; set; }`

Данное свойство определяет, каким образом значения ячеек данной колонки должны обрабатываться преобразователем (`INodeColumnTransform`), назначенным данной колонке.

`Int32 Width { get; set; }`

Возвращает или устанавливает ширину колонки в пикселах.

8.19.1.4 Методы класса:

NodeColumn (конструктор)

Конструктор, позволяющий создать колонку.

Синтаксис:

```
void NodeColumn (object source)
void NodeColumn (System.Guid schemeGuid; object id;
System.Type dataType; Intermech.FieldTypes attrType; string caption)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType;
Intermech.FieldTypes attrType; string caption; Intermech.Kernel.Search.ColumnContents contents)
void NodeColumn (System.Guid schemeGuid; object id;
System.Type dataType; Intermech.FieldTypes attrType; string caption; string shortCaption; string hint)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType; Intermech.FieldTypes attrType;
string caption; Intermech.Navigator.Interfaces.NodeColumnSortOrder sortOrder; Int32 sortIndex)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType;
Intermech.FieldTypes attrType; string caption;
Intermech.Navigator.Interfaces.NodeColumnSortOrder sortOrder; Int32 sortIndex;
Intermech.Kernel.Search.ColumnContents contents)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType;
Intermech.FieldTypes attrType; string caption;
Intermech.Navigator.Interfaces.NodeColumnSortOrder sortOrder; Int32 sortIndex;
string shortCaption; string hint)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType;
Intermech.FieldTypes attrType; string caption;
Intermech.Navigator.Interfaces.NodeColumnSortOrder sortOrder; Int32 sortIndex;
string shortCaption; string hint; Intermech.Kernel.Search.ColumnContents contents)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType;
Intermech.FieldTypes attrType; string caption;
Intermech.Navigator.Interfaces.NodeColumnSortOrder sortOrder; Int32 sortIndex;
string shortCaption; string hint; bool systemAttr)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType;
Intermech.FieldTypes attrType; string caption;
Intermech.Navigator.Interfaces.NodeColumnSortOrder sortOrder; Int32 sortIndex;
string shortCaption; string hint; bool systemAttr;
Intermech.Kernel.Search.ColumnContents contents)
void NodeColumn (System.Guid schemeGuid; object id; System.Type dataType;
Intermech.FieldTypes attrType; string caption; string shortCaption; string hint;
Intermech.Kernel.Search.ColumnContents contents)
```

Параметры:

schemeGuid

Идентификатор схемы колонки.

id

Идентификатор колонки.

dataType

Тип данных, содержащихся в колонке.

attrType

Тип данных атрибута, связанного с колонкой.

caption

Заголовок колонки.

contents

Содержимое ячеек колонки.

sortOrder

Направление сортировки.

sortIndex

Номер колонки в списке отсортированных колонок.

shortCaption

Краткое наименование колонки.

hint

Подсказка.

systemAttr

Является ли атрибут колонки системным.

source

Объект-источник.

Assign

Скопировать в текущий объект поля из другого объекта.

Синтаксис:

void Assign (object source)

Параметры:

source

Объект-источник.

Clear

Очистить поля класса.

Синтаксис:

void Clear ()

Clone

Создать копию объекта, идентичную натуральному.

Синтаксис:

object Clone ()

Возвращаемое значение:

Копия объекта, идентичная натуральному.

CompareTo

Сравнить колонку атрибута с указанным объектом.

Синтаксис:

Int32 CompareTo (Intermech.Navigator.Interfaces.NodeColumn other)

Параметры:

other

Объект для сравнения.

Возвращаемое значение:

-1, 0, 1.

CompareTo

Сравнить колонку атрибута с указанным объектом.

Синтаксис:

Int32 CompareTo (object obj)

Параметры:

obj

Объект для сравнения.

Возвращаемое значение:

-1, 0, 1.

Equals

Сравнить текущий экземпляр колонки атрибута с указанным объектом.

Синтаксис:

bool Equals (object obj)

Параметры:

obj

Объект для сравнения.

Возвращаемое значение:

true, если текущий экземпляр колонки атрибута равен указанному объекту.

FullEquals

Сравнить текущий экземпляр колонки атрибута с указанным объектом.

Синтаксис:

```
bool FullEquals (object obj)
```

Параметры:

obj

Объект для сравнения.

Возвращаемое значение:

true, если текущий экземпляр колонки атрибута равен указанному объекту.

GetHashCode

Рассчитать 32-битный хэш-код для текущего экземпляра класса.

Синтаксис:

```
Int32 GetHashCode ()
```

Возвращаемое значение:

32-битный хэш-код для текущего экземпляра класса.

ToString

Вернуть строковое представление колонки.

Синтаксис:

```
string ToString ()
```

Возвращаемое значение:

Строковое представление колонки.

Рассмотрим коллекцию колонок `NodeColumn` – класс типа `NodeColumnCollection`.

8.19.2 Класс `NodeColumnCollection`

Коллекция предназначена для хранения и управления списком колонок пространства навигации `NodeColumn`. Класс унаследован от типизированного списка `List[NodeColumn]`, реализует интерфейс `ICloneable`.

8.19.2.1 Пространство имен

Intermech.Navigator.Interfaces

8.19.2.2 Реализуемые интерфейсы:

System.ICloneable

8.19.2.3 Методы класса:

`NodeColumnCollection` (конструктор)

Создать коллекцию колонок.

Синтаксис:

```
void NodeColumnCollection ()
```

```
void NodeColumnCollection (System.Collections.Generic.IEnumerable<NodeColumn> collection)
```

```
void NodeColumnCollection (Int32 capacity)
```

```
void NodeColumnCollection (object source)
```

Параметры:

collection

Коллекция-прототип.

capacity

Ёмкость коллекции.

source

Объект-источник.

Add

Добавить колонку в коллекцию.

Синтаксис:

```
void Add (Intermech.Navigator.Interfaces.NodeColumn item)
```

```
void Add (Intermech.Navigator.Interfaces.NodeColumn column; Int32 width)
```

Параметры:

item

Колонка.

column

Колонка.

width

Новая ширина колонки.

Assign

Скопировать в текущий объект поля из другого объекта.

Синтаксис:

```
void Assign (object source)
```

Параметры:

source

Объект-источник.

Clear

Очистить поля класса.

Синтаксис:

```
void Clear ()
```

Clone

Создать копию объекта, идентичную натуральной.

Синтаксис:

```
object Clone ()
```

Возвращаемое значение:

Копия объекта, идентичная натуральной.

ColumnIDExists

Проверяет, есть ли в коллекции колонка с указанным идентификатором, либо любая из колонок с перечисленными идентификаторами.

Синтаксис:

```
bool ColumnIDExists (object columnID)
```

```
bool ColumnIDExists (object columnID; System.Guid schemeGuid)
```

Параметры:

columnID

Идентификатор колонки.

schemeGuid

Идентификатор схемы.

Возвращаемое значение:

true, если колонка с указанным идентификатором найдена.

ColumnIDsExists

Проверяет, есть ли в коллекции колонка с указанным идентификатором, либо любая из колонок с перечисленными идентификаторами.

Синтаксис:

```
bool ColumnIDsExists (System.Collections.IList columnIDs)
```

```
bool ColumnIDsExists (System.Collections.Generic.IList<Int32> columnIDs)
```

Параметры:

columnIDs

Идентификаторы колонок.

Возвращаемое значение:

true, если колонка с любым из указанных идентификаторов найдена.

Contains

Проверить наличие колонки в коллекции.

Синтаксис:

```
bool Contains (Intermech.Navigator.Interfaces.NodeColumn item)
```

Параметры:

item

Искомая колонка.

Возвращаемое значение:

true, если найдена.

Find

Отыскать колонку по её идентификатору.

Синтаксис:

```
Intermech.Navigator.Interfaces.NodeColumn Find (object columnID)
```

```
Intermech.Navigator.Interfaces.NodeColumn Find (object columnID; System.Guid schemeGuid)
```

```
Intermech.Navigator.Interfaces.NodeColumn Find (string key)
```

Параметры:

columnID

Идентификатор колонки.

schemeGuid

Идентификатор схемы.

key

Ключ колонки.

Возвращаемое значение:

Колонка или null.

HasSortedColumns

Есть ли в коллекции хотя бы одна сортируемая колонка.

Синтаксис:

```
bool HasSortedColumns ()
```

Возвращаемое значение:

true, если в коллекции есть хотя бы одна сортируемая колонка.

Remove

Удалить указанную колонку из коллекции.

Синтаксис:

```
bool Remove (Intermech.Navigator.Interfaces.NodeColumn item)
```

Параметры:

item

Удаляемая колонка.

Возвращаемое значение:

true, если колонка была удалена.

RemoveAt

Удалить колонку с указанным индексом из коллекции.

Синтаксис:

```
void RemoveAt (Int32 index)
```

Параметры:

index

Индекс удаляемой колонки.

RemoveNonSortedColumns

Удалить из коллекции колонки, которые не сортируются, проверить у остальных свойство SortIndex.

Синтаксис:

```
void RemoveNonSortedColumns ()
```

RemoveSortInfo

Удалить у всех колонок информацию о сортировке (все колонки становятся несортируемыми).

Синтаксис:

```
void RemoveSortInfo ()
```

Sort

Выполнить сортировку.

Синтаксис:

```
void Sort (bool ascending)
```

Параметры:

ascending

true - по возрастанию.

SortByIndex

Упорядочить колонки в коллекции согласно их свойству SortIndex - очередности сортировки.

Синтаксис:

```
void SortByIndex ()
```

SortBySchemesAndNames

Упорядочить колонки в коллекции согласно названиям схем, а затем по названию колонок.

Синтаксис:

```
void SortBySchemesAndNames ()
```

8.19.2.4 Статические методы класса:

CorrectSortIndex

Если из коллекции колонок были удалены колонки, участвующие в сортировке, то, вызвав данный метод, можно откорректировать значения свойств SortIndex у оставшихся сортируемых колонок, причём эти значения будут корректно упорядочены по возрастанию.

Синтаксис:

```
static void CorrectSortIndex (Intermech.Navigator.Interfaces.NodeColumnCollection columns)
```

Параметры:

columns

Список колонок навигатора, у которых следует откорректировать значения свойства SortIndex.

GetSortedColumns

Метод выбирает из указанной коллекции все сортируемые колонки и возвращает новую коллекцию колонок, в которую входят сортируемые колонки, упорядоченные по возрастанию их свойства SortIndex.

Синтаксис:

```
static Intermech.Navigator.Interfaces.NodeColumnCollection GetSortedColumns  
(Intermech.Navigator.Interfaces.NodeColumnCollection columns)
```

Параметры:

columns

Список колонок навигатора.

Возвращаемое значение:

Упорядоченная коллекция сортируемых колонок.

8.19.3 Перечислитель NodeColumnSortOrder

Возможные направления сортировки данных в виртуальных колонках "Навигатора".

8.19.3.1 Пространство имен

Intermech.Navigator.Interfaces

8.19.3.2 Значения перечислителя:

Ascending = 0x00000001

Содержимое колонки сортируется по возрастанию.

Descending = 0x00000002

Содержимое колонки сортируется по убыванию.

None = 0x00000000

Колонка не участвует в сортировке.

8.19.4 Перечислитель CellTransformationMode

Перечислитель указывает, каким образом будут преобразовываться ячейки в колонке.

8.19.4.1 Пространство имен

Intermech.Navigator.Interfaces

8.19.4.2 Значения перечислителя:

ConvertToCellValue = 0x00000002

Значения будут преобразованы в составной класс - [Оригинальное значение] + [Значение для отображения на экране].

ConvertToString = 0x00000001

Значения будут преобразованы в строки для отображения на экране.

WithoutTransformation = 0x00000000

Значения преобразовываться не будут.

8.19.5 Перечислитель ContentType

Данный перечислитель (набор битовых флажков, [Flags]) позволяет указать, какое содержимое находится в составе элемента пространства навигации.

8.19.5.1 Пространство имен

Intermech.Navigator.Interfaces

8.19.5.2 Значения перечислителя:

Folders = 0x00000001

В содержимое узла могут входить папки.

Hidden = 0x00000004

В содержимое узла могут скрытые узлы.

None = 0x00000000

Содержимое узла неизвестно.

NonFolders = 0x00000002

В содержимое узла могут входить не папки (аналог файлов в файловой системе).

8.19.6 Перечислитель NodeQueryOptions

8.19.6.1 Пространство имён

Intermech.Navigator.Interfaces

Перечислитель (набор битовых флажков, [Flags]) позволяет задать некоторые опции для интерфейса-запроса к источнику данных INodeQuery.

8.19.6.2 Значения перечислителя

None = 0

Нет никаких опций.

ReceiveTotalRecordsCount = 1

Перед первым пакетным чтением определить, сколько всего записей может быть получено.

8.19.7 Интерфейс INodeColumnScheme

Описывает схему виртуальных колонок, используемых в навигаторе. Схема отвечает за создание, сериализацию и десериализацию колонок, а также предоставляет для колонок преобразование по умолчанию, которое может использоваться для вывода значений колонок в гридах.

Схема виртуальных колонок Навигатора INodeColumnScheme позволяет выполнять следующие действия: – создаёт колонки; – выполняет их сериализацию/десериализацию; – отыскивает для колонки интерфейс преобразования значений ячеек INodeColumnTransform.

Внимание! Класс, реализующий данный интерфейс, должен быть потокобезопасным.

8.19.7.1 Пространство имен

Intermech.Navigator.Interfaces

8.19.7.2 Свойства интерфейса:

string Name { get; }

Возвращает название схемы колонок, которое выводится в диалоге настройки отображения колонок.

8.19.7.3 Методы интерфейса:

ColumnIDToPersistName

Возвращает постоянное имя колонки, которое можно использовать для долговременного хранения (т.е. между сеансами работы универсального клиента).

Синтаксис:

string ColumnIDToPersistName (object columnID)

Параметры:

columnID

Идентификатор виртуальной колонки.

Возвращаемое значение:

Постоянное имя колонки.

CreateColumn

Создает виртуальную колонку без сортировки по указанному идентификатору. Если колонки с заданным идентификатором в схеме нет - то метод вернет null.

Синтаксис:

Intermech.Navigator.Interfaces.NodeColumn CreateColumn (System.Guid schemeGuid; object columnID)

Параметры:

schemeGuid

Guid схемы.

columnID

Идентификатор виртуальной колонки.

Возвращаемое значение:

Виртуальная колонка.

CreateColumn

Создает виртуальную колонку с заданным направлением сортировки по указанному идентификатору. Если колонки с таким идентификатором в схеме нет - то метод вернет null.

Синтаксис:

Intermech.Navigator.Interfaces.NodeColumn CreateColumn (System.Guid schemeGuid; object columnID;
Intermech.Navigator.Interfaces.NodeColumnSortOrder sortOrder; Int32 sortIndex)

Параметры:

schemeGuid

Guid схемы.

columnID

Идентификатор виртуальной колонки.

sortOrder

Направление сортировки.

sortIndex

Очередность сортировки (-1 - не сортируется).

Возвращаемое значение:

Виртуальная колонка.

GetDefaultTransform

Возвращает преобразование по умолчанию для указанной виртуальной колонки. Если преобразование не задано, то метод вернет null.

Синтаксис:

Intermech.Navigator.Interfaces.INodeColumnTransform GetDefaultTransform (object columnID)

Параметры:

columnID

Идентификатор виртуальной колонки.

Возвращаемое значение:

Преобразование по умолчанию.

PersistNameToColumnID

Восстанавливает идентификатор виртуальной колонки по ее постоянному имени, которое действительно только на текущий сеанс работы универсального клиента. Если восстанавливаемая колонка не существует, то метод должен вернуть null.

Синтаксис:

object PersistNameToColumnID (string persistName)

Параметры:

persistName

Постоянное имя колонки.
Возвращаемое значение:
Идентификатор виртуальной колонки.

8.19.8 Интерфейс `INodeColumnTransform`

Данный интерфейс предназначен для преобразования значений колонок в значения для отображения на экране, например, для расшифровки идентификаторов пользователей в их имена и т.п.

8.19.8.1 Пространство имён

[Intermech.Navigator.Interfaces](#)

Внимание! Класс, реализующий данный интерфейс, должен быть потокобезопасным.

8.19.8.2 Свойства интерфейса

`Type DataType { get; }`

Возвращает тип значения, которое будет получено после выполнения преобразования.

8.19.8.3 Методы интерфейса

[Apply](#)

Метод выполняет преобразование исходного значения колонки в новое значение, если оно требуется какими-либо правилами. Если содержимое колонки `column.Contetns` отлично от значения по умолчанию `Text`, либо у колонки метод трансформации задан как `CellTransformationMode.ConvertToCellValue`, то преобразование вернёт значение в виде экземпляра класса `CellValue`, в котором хранятся одновременно два значения – оригинальное и новое значения.

Синтаксис:

```
object Apply (object sourceValue, NodeColumn column);
```

Параметры:

`sourceValue`

Исходное значение колонки.

`column`

Колонка, для которой выполняется преобразование.

Результат:

Преобразованное или исходное значение.

В ядре клиентского приложения реализованы два дополнительных преобразователя – `IListAttributeTransform` и `IBoolAttributeTransform`.

Класс, реализующий интерфейс `IListAttributeTransform`, выполняет преобразование значений всех списковых атрибутов в значения для отображения на экране.

Класс, реализующий интерфейс `IBoolAttributeTransform`, выполняет преобразование всех булевых атрибутов в значения «Да»/«Нет».

8.19.9 Класс `CellValue`

Данный класс предназначен для хранения пары значений ячейки – оригинального значения, полученного у источника данных, и преобразованного значения для отображения на экране.

Экземпляры данного класса создаются интерфейсом `INodeColumnTransform`. Экземпляры данного класса реализуют все операторы для явного (`explicit`) и неявного (`implicit`) преобразования типов своих значений в стандартные типы .NET Framework, а также реализуют интерфейсы `ICloneable`, `IConvertible` и `IComparable`.

8.19.9.1 Пространство имён

[Intermech.Navigator.Interfaces](#)

8.19.9.2 Свойства класса

`object Caption { get; set; }`

Значение ячейки для отображения на экране.

`object Value { get; set; }`

Оригинальное значение ячейки, полученное из источника данных.

8.19.9.3 Статические методы класса

[GetValue](#)

Метод анализирует значение `column.Contents`, если оно равно `Text`, возвращает значение `newValue`, иначе возвращает экземпляр класса `CellValue(sourceValue, newValue)`.

Синтаксис:

```
object GetValue (object sourceValue, NodeColumn column, object newValue);
```

Параметры:

`sourceValue`

Оригинальное значение, полученное из источника данных.

`nodeColumn`

Колонка «Навигатора», для которой выполняется преобразование значения.

`newValue`

Новое значение, которое сгенерировано интерфейсом `INodeColumnTransform`.

Результат:

Преобразованное или оригинальное значение.

8.19.10 Интерфейс `IDescriptor`

Данный интерфейс описывает элемент пространства навигации, причём данный элемент может быть корневым (находиться в корне дерева навигации).

Интерфейс унаследован от интерфейса `INodeItems`.

8.19.10.1 Пространство имён

`Intermech.Navigator.Interfaces`

Интерфейс, который должен реализовываться классами, описывающими различные типы элементов из пространства навигации, которые можно включать в состав различных виртуальных элементов (например, в корень всего дерева **Навигатора**).

8.19.10.2 Методы интерфейса

`GetRecordNodeID`

Метод возвращает актуальный на момент вызова метода идентификатор объекта, который описывается дескриптором. Метод может возвращать **null**, если объект не доступен или не существует.

Синтаксис:

```
INodeID GetRecordNodeID ();
```

Результат:

Идентификатор элемента пространства навигации, который описан данным интерфейсом.

`GetRecordValues`

Метод возвращает значения полей для объекта, описываемого унифицированным дескриптором. Метод может возвращать **null**, если объект не доступен или не существует.

Синтаксис:

```
object[] GetRecordValues (INodeID nodeID, object[] fields);
```

Параметры:

`nodeID`

Идентификатор элемента пространства навигации.

`fields`

Коллекция полей (атрибутов), данные для которых требуется получить.

Результат:

Значения полей для объекта.

`MapColumnToField`

Метод возвращает идентификатор поля источника данных для указанной виртуальной колонки. Если данная колонка не поддерживается, то метод возвращает **null**.

Синтаксис:

```
object MapColumnToField (NodeColumn column);
```

Параметры:

`column`

Колонка **Навигатора**.

Результат:

Идентификатор поля источника данных (например, атрибута) или **null**.

Для различных задач в ядре универсального клиента существует несколько стандартных классов, реализующих IDescriptor:

Пространство имён	Назначение дескриптора
Intermech.Navigator.DBOBJECTS.Descriptor	Предназначен для создания элементов пространства навигации «Объект базы данных»
Intermech.Navigator.DBOBJECTTYPES.Descriptor	Предназначен для создания элементов пространства навигации «Тип объекта базы данных»
Intermech.Navigator.VirtualNodes.HiveDescriptor	Предназначен для создания всевозможных виртуальных элементов пространства навигации
Intermech.Navigator.CustomNode.Descriptor	Составной дескриптор – состоит из коллекции дескрипторов типа HiveDescriptor – для создания составных виртуальных элементов пространства навигации
Intermech.Navigator.DBOBJECTTYPES.HiveDescriptor	Дескриптор корневого элемента дерева типа объектов – создаёт корневой узел «Объекты»
Intermech.Navigator.DBOBJECTS.AdvListDescriptor	Дескриптор, позволяющий создавать иерархии элементов вида [Список объектов] => [Список для каждого объекта]
Intermech.Navigator.GlobalNode.Descriptor	Дескриптор, создающий корневой элемент всего пространства навигации Информационное пространство
Intermech.Navigator.Selections.HiveDescriptor	Дескриптор, создающий элементы типа «Выборка» и «Классификаторы», привязанные к определённым родительским элементам пространства навигации

Рассмотрим подробнее интерфейс INodeItems, от которого унаследован интерфейс IDescriptor.

8.19.11 Интерфейс INodeItems

Классы, реализующие данный интерфейс, позволяют работать с дочерними элементами пространства навигации по их идентификаторам INodeID.

8.19.11.1 Пространство имён

Intermech.Navigator.Interfaces

Интерфейс позволяет получать различного рода информацию о дочерних элементах пространства навигации по их идентификаторам INodeID.

8.19.11.2 Методы интерфейса

Deserialize

Метод восстанавливает идентификатор элемента пространства навигации из его сериализованного представления.

Синтаксис:

```
INodeID Deserialize (PersistentState persistNodeID);
```

Параметры:

persistNodeID

Сериализованное представление элемента пространства навигации.

Результат:

Восстановленный идентификатор элемента пространства навигации.

GetAddress

Метод возвращает адрес указанного элемента. Данный адрес можно использовать в адресной строке **Навигатора** для доступа к объекту.

Синтаксис:

```
string GetAddress (INodeID nodeID);
```

Параметры:

nodeID

Идентификатор элемента пространства навигации.

Результат:

Адрес указанного элемента.

GetAnalyser

Метод возвращает интерфейс анализатора, который позволяет указанному элементу пространства навигации обработать какие-либо уведомления от универсального клиента.

Синтаксис:

```
IUpdateAnalyser GetAnalyser (NodeViewCapabilities capabilities, object sender, NotificationEventArgs e);
```

Параметры:

capabilities

Сведения о возможностях элемента пространства навигации.

sender

Объект, отправивший уведомление.

e

Аргументы уведомления.

Результат:

Анализатор или **null**, если анализатор не найден.

GetAttributesOf

Метод возвращает набор атрибутов указанного элемента пространства навигации.

Синтаксис:

```
ContentAttributes GetAttributesOf (INodeID nodeID);
```

Параметры:

nodeID

Идентификатор элемента пространства навигации.

Результат:

Набор атрибутов.

GetChild

Метод возвращает интерфейс элемента пространства навигации по его идентификатору.

Синтаксис:

```
INode GetChild (INodeID nodeID);
```

Параметры:

nodeID

Идентификатор элемента пространства навигации.

Результат:

Элемент пространства навигации.

GetData

Метод возвращает данные указанного формата, которые может вернуть указанный элемент пространства навигации.

Синтаксис:

```
object GetData (INodeID nodeID, Type dataFormat);
```

Параметры:

nodeID

Идентификатор элемента пространства навигации.

dataFormat

Тип запрашиваемых данных.

Результат:

Данные запрошенного формата или **null**.

GetData

Метод возвращает данные указанного формата, которые может вернуть каждый из указанных элементов пространства навигации.

Синтаксис:

```
object[] GetData (NodeIDCollection nodeIDs, Type dataFormat);
```

Параметры:

nodeIDs

Коллекция идентификаторов элементов пространства навигации.

dataFormat

Тип запрашиваемых данных.

Результат:

Данные запрошенного формата для каждого элемента из коллекции.

GetService

Метод возвращает сервис указанного типа.

Синтаксис:

```
object GetService (Type service);
```

Параметры:

service

Тип запрашиваемого сервиса.

Результат:

Сервис или **null**.

ParseAddress

Метод восстанавливает идентификатор дочернего элемента пространства навигации по его адресу.

Синтаксис:

```
INodeID ParseAddress (string address);
```

Параметры:

address

Адрес дочернего элемента пространства навигации.

Результат:

Идентификатор дочернего элемента или **null**.

Serialize

Метод выполняет сериализацию указанного идентификатора элемента пространства навигации в специальный класс.

Синтаксис:

```
PersistentState Serialize (INodeID nodeID);
```

Параметры:

nodeID

Сериализуемый идентификатор элемента пространства навигации.

Результат:

Сериализованное представление указанного элемента пространства навигации.

8.19.12 Интерфейс INode

Данный интерфейс является основой всех элементов пространства навигации. Данный интерфейс унаследован от интерфейса INodeItems. Кроме функционала INodeItems, данный интерфейс позволяет управлять дочерними элементами, а также получать ссылку на интерфейс INodeQuery – интерфейс, который позволяет получать информацию о дочерних элементах из источника данных.

В составе элемента пространства навигации могут входить элементы нескольких разновидностей, а именно папки, не папки (аналоги файлов в файловой системе) и скрытые узлы.

8.19.12.1 Пространство имён

Intermech.Navigator.Interfaces

Внимание! Данный интерфейс, связанный с каким-либо элементом пространства навигации, работает с данными его дочерних элементов.

8.19.12.2 Методы интерфейса

GetDefaultColumns

Метод возвращает коллекцию колонок по умолчанию для узла с указанным содержанием.

Синтаксис:

```
NodeColumnCollection GetDefaultColumns (ContentType content);
```

Параметры:

content

Содержимое узла (папки, не папки, скрытые узлы), для которых требуется получить коллекцию колонок по умолчанию.

Результат:

Коллекция колонок по умолчанию для указанного содержимого узла.

GetQuery

Метод возвращает ссылку на интерфейс объекта-запроса, с помощью которого выполняется чтение списка и данных для дочерних элементов текущего элемента пространства навигации. Если у данного элемента нет дочерних элементов, данный метод должен вернуть **null**.

Синтаксис:

```
INodeQuery GetQuery (ContentType content);
```

Параметры:

content

Содержимое узла (папки, не папки, скрытые узлы), для которых требуется получить интерфейс для запроса к источнику данных.

Результат:

Интерфейс для запроса к источнику данных или **null**.

GetSupportedColumns

Метод возвращает коллекцию допустимых колонок для узла с указанным содержанием.

Синтаксис:

```
NodeColumnCollection GetSupportedColumns (ContentType content, string ColumnSetName);
```

Параметры:

content

Содержимое узла (папки, не папки, скрытые узлы), для которых требуется получить коллекцию допустимых колонок.

ColumnSetName

Название набора колонок. Если указать **string.Empty**, то будет получена стандартная коллекция допустимых колонок.

Результат:

Коллекция допустимых колонок для указанного содержимого узла.

GetSupportedColumnSetNames

Метод возвращает допустимые названия наборов колонок. Если вернёт **null**, то допустимо только название **string.empty**.

Синтаксис:

```
List<string> GetSupportedColumnSetNames ();
```

Результат:

Допустимые названия наборов колонок или **null**.

Refresh

Метод обновляет внутренние структуры элемента пространства навигации.

Синтаксис:

```
void Refresh ();
```

В ядре универсального клиента реализовано несколько стандартных классов, которые реализуют интерфейс **INode**, и предназначены для наиболее типичных задач:

Пространство имён	Назначение класса
Intermech.Navigator.Parts.CompositeNode	Базовый класс для создания элементов пространства навигации. Данный узел состоит из частей – папок и не папок. Папки отображаются в дереве, не папки – в закладках.
Intermech.Navigator.DBObjectTypes.ObjectTypesNode	Класс реализует узел «Все объекты» - корень дерева «Навигатора» для отображения всех типов объектов базы данных

Пространство имён	Назначение класса
Intermech.Navigator.DBObjectTypes.ObjectTypeNode	Класс реализует узел «Тип объекта» - узел из списка типов объектов
Intermech.Navigator.DBObjects.ObjectNode	Класс реализует узел, представляющий собой объект базы данных. качестве папок возвращает объекты, входящие в состав обрабатываемого элементом объекта связью по умолчанию. Не-папок у этого элемента нет.
Intermech.Navigator.DBObjects.AdvObjectsListNode	Класс реализует узел, позволяющий отображать списки объектов вида [Список объектов] => [Список для каждого объекта].

8.19.13 Интерфейс IContextAware

Каждый элемент пространства навигации может работать в каком-то контексте. Контекст – это контейнер сервисов, к которым может обращаться элемент. В зависимости от наличия или отсутствия этих служб, а также от их результатов работы их методов или значений их свойств поведение элемента пространства навигации может очень сильно меняться.

8.19.13.1 Пространство имён

Intermech.Navigator.Interfaces.IContextAware

Если данный интерфейс реализован у элемента пространства навигации, то элемент автоматически будет получать текущий контекст от своего родительского элемента.

8.19.13.2 Свойства интерфейса

IServiceProvider Services { get; set; }

Контейнер сервисов, который является контекстом для элемента пространства навигации.

8.19.14 Интерфейс INodeID

Данный интерфейс описывает унифицированный идентификатор любого элемента пространства навигации.

8.19.14.1 Пространство имён

Intermech.Navigator.Interfaces

Все интерфейсы **Навигатора** используют данный интерфейс как точку отсчёта.

8.19.14.2 Свойства интерфейса

Int32 CategoryID { get; }

Идентификатор категории элемента пространства навигации.

Внимание! Каждый элемент обязан принадлежать какой-то категории.

Int32TypeID { get; }

Идентификатор типа элемента пространства навигации внутри категории.

В ядре универсального клиента реализовано несколько стандартных классов, которые поддерживают интерфейс INodeID, и предназначены для некоторых наиболее типичных задач:

Пространство имён	Назначение класса
Intermech.Navigator.DBObjects.NodeID	Предназначен для создания идентификаторов элементов пространства навигации «Объект базы данных»
Intermech.Navigator.DBObjectTypes.Implementation.NodeID	Предназначен для создания идентификаторов элементов пространства навигации «Тип объекта базы данных»
Intermech.Navigator.EventLog.NodeID	Предназначен для создания идентификаторов элементов пространства навигации для журнала событий
Intermech.Navigator.VirtualNodes.HiveNodeID	Предназначен для создания идентификаторов виртуальных элементов пространства навигации

8.19.15 Класс NodeIDPath

Дерево **Навигатора** – это иерархия элементов пространства навигации. У каждого элемента есть свой путь, по которому можно отыскать данный элемент, сохранить, а затем восстановить сфокусированный элемент в дереве и т.п. Класс NodeIDPath позволяет сохранить полный путь к любому элементу пространства навигации. Он основан на коллекции унифицированных идентификаторов INodeID, а также содержит в своём составе описание корневого элемента пространства навигации IDescriptor, по которому было построено дерево пространства навигации.

8.19.15.1 Пространство имён

Intermech.Navigator.Interfaces.NodeIDPath

Путь к элементу пространства навигации. Путь основан на дескрипторе корневого узла и массиве унифицированных идентификаторов родительских элементов. Класс реализует интерфейсы IEnumerable и ICloneable.

8.19.15.2 Конструкторы

Синтаксис:

```
NodeIDPath (IDescriptor rootDescriptor);
```

Параметры:

rootDescriptor

Описание корневого элемента пространства навигации.

Результат:

Создаётся экземпляр класса с пустым путём, но с указанным описанием корневого элемента пространства навигации.

Синтаксис:

```
NodeIDPath (NodeIDPath path);
```

Параметры:

path

Путь к элементу пространства навигации, копия которого будет размещена в создаваемом пути.

Результат:

Создаётся экземпляр класса с путём, скопированным у указанного пути.

Синтаксис:

```
NodeIDPath (NodeIDPath path, INodeID nodeID);
```

Параметры:

path

Путь к элементу пространства навигации, копия которого будет размещена в создаваемом пути.

nodeID

Идентификатор элемента пространства навигации, который будет добавлен к скопированному пути.

Результат:

Создаётся экземпляр класса с путём, скопированным у указанного пути; кроме того, к пути будет добавлен указанный идентификатор элемента пространства навигации.

8.19.15.3 Свойства класса

```
INodeID FirstID { get; }
```

Унифицированный идентификатор первого элемента пространства навигации из пути или null, если путь пуст.

```
INodeID LastID { get; }
```

Унифицированный идентификатор последнего элемента пространства навигации из пути или null, если путь пуст.

```
Int32 Length { get; }
```

Количество идентификаторов элементов пространства навигации, которое содержится в пути.

```
IDescriptor RootDescriptor { get; set; }
```

Описание корневого элемента пространства навигации.

```
INodeID this[Int32 Index] { get; }
```

Получить унифицированный идентификатор элемента пространства навигации с указанным индексом.

8.19.15.4 Методы класса

Add

Метод добавляет указанный идентификатор элемента пространства навигации в путь.

Синтаксис:

```
void Add (INodeID NodeID);  
void Add (INodeID NodeID, bool ToEnd);
```

Параметры:

NodeID

Добавляемый идентификатор элемента пространства навигации.

ToEnd

Если указать **true** - добавить описание узла в конец пути, иначе вставить его в начало пути.

Clear

Метод очищает путь, но при этом сохраняет значение описания корневого элемента.

Синтаксис:

```
void Clear ();
```

8.19.16 Класс PartSlot

Каждый элемент пространства навигации является составным элементом. Его компонентами могут быть папки, не папки (аналог файлов) и скрытые узлы. Каждый из таких компонентов представлен классом, реализующим интерфейс INodePart, а хранятся такие части в узлах пространства навигации в виде коллекции типизированных слотов – экземпляров класса **PartSlot : Slot<INodePart>**.

8.19.16.1 Пространство имён

Intermech.Navigator.Parts

Слот, в котором хранится ссылка на интерфейс типа INodePart. Экземпляры данного класса создаются в методах CreateFolderSlots и CreateNonFolderSlots классов, реализующих интерфейсы INode.

8.19.16.2 Конструктор

Синтаксис:

```
PartSlot (Guid partGuid, INodePart part);
```

Параметры:

partGuid

Уникальный идентификатор части элемента пространства навигации.

part

Часть элемента пространства навигации.

Результат:

Создаётся заполненный экземпляр класса.

8.19.16.3 Свойства класса

```
Guid Guid { get; }
```

Уникальный идентификатор части элемента пространства навигации, хранящейся в слоте.

8.19.17 Интерфейс INodePart

Интерфейс унаследован от INodeItems, и позволяет работать с дочерними элементами пространства навигации. Дополнительно интерфейс позволяет работать с источниками данных, а также управлять коллекциями допустимых колонок и колонок по умолчанию.

8.19.17.1 Пространство имён

Intermech.Navigator.Parts

Из данных интерфейсов формируется список частей элемента пространства навигации (в виде List<PartSlot>).

8.19.17.2 Свойства интерфейса

```
object Owner { get; set; }
```

Ссылка на объект, в состав которого входит данный интерфейс.

8.19.17.3 Методы интерфейса

GetDefaultColumns

Метод возвращает коллекцию колонок по умолчанию для данной части элемента пространства навигации.

Синтаксис:

```
NodeColumnCollection GetDefaultColumns ();
```

Результат:

Коллекция колонок по умолчанию.

GetQuery

Метод возвращает интерфейс объекта-запроса к источнику данных, используемого для чтения содержимого элементов из пространства навигации.

Синтаксис:

```
NodeColumnCollection GetQuery ();
```

Результат:

Интерфейс объекта-запроса к источнику данных.

GetSupportedColumns

Метод возвращает коллекцию допустимых колонок с указанным именем для данной части элемента пространства навигации.

Синтаксис:

```
NodeColumnCollection GetSupportedColumns (string ColumnSetName);
```

Параметры:

ColumnSetName

Название требуемого набора колонок или **string.empty**.

Результат:

Коллекция допустимых колонок.

GetSupportedColumnSetNames

Метод возвращает список названий допустимых наборов колонок или **null**.

Синтаксис:

```
List<string> GetSupportedColumnSetNames ();
```

Результат:

Список допустимых названий допустимых наборов колонок. Если будет возвращено значение **null**, то есть только одно допустимое название набора колонок – **string.empty**.

8.19.18 Интерфейс INodeQuerySupport

8.19.18.1 Пространство имён

Intermech.Navigator.Queries

Интерфейс помогает подготовить запрос к источнику данных к выполнению, а затем – обработать полученные результаты.

8.19.18.2 Методы интерфейса

CreateNodeId

Метод создаёт унифицированный идентификатор элемента пространства навигации на основании значений, полученных из источника данных.

Синтаксис:

```
INodeID CreateNodeId (object[] fieldValues, RecordAdapter adapter);
```

Параметры:

fieldVallues

Значения полей, полученные от источника данных.

adapter

Адаптер полей источника данных.

Результат:

Унифицированный идентификатор элемента пространства навигации.

CreateRecordId

Метод создаёт и возвращает идентификатор элемента пространства навигации в источнике данных по его унифицированному идентификатору.

Синтаксис:

```
object CreateRecordId (INodeID nodeID);
```

Параметры:

nodeID

Унифицированный идентификатор элемента пространства навигации.

Результат:

Идентификатор элемента пространства навигации в источнике данных. Например, идентификатор атрибута, название атрибута.

GetSpecialFields

Метод создаёт и возвращает список идентификаторов полей в источнике данных, которые обязательно требуется получить у источника в результате выполнения запроса.

Синтаксис:

```
list<object> GetSpecialFields ();
```

Результат:

Список идентификаторов полей в источнике данных.

MapColumnToField

Метод возвращает идентификатор поля источника данных для указанной виртуальной колонки. Если данная колонка не поддерживается, то метод должен вернуть **null**.

Синтаксис:

```
object MapColumnToField (NodeColumn column);
```

Параметры:

column

Унифицированный идентификатор элемента пространства навигации.

Результат:

Идентификатор поля источника данных для указанной виртуальной колонки.

8.19.19 Интерфейс INodeQuery

8.19.19.1 Пространство имён

Intermech.Navigator.Interfaces

Интерфейс объекта-запроса к источнику данных, используемого для чтения содержимого элементов из пространства навигации.

8.19.19.2 Свойства интерфейса

```
object Bookmark { get; }
```

Возвращает закладку, определяющую позицию для чтения следующей порции дочерних элементов или **null**, если была прочитана последняя порция. Данное свойство используется при пакетном чтении данных.

```
NodeQueryOptions Options { get; set; }
```

Набор дополнительных условий для выполнения запроса к источнику данных.

```
Int32 RecordCount { get; }
```

Возвращает количество прочитанных в результате выполнения запроса дочерних элементов.

```
Int64 TotalRecordCount { get; }
```

Возвращает количество всех элементов, которые могут быть получены с помощью данного запроса. Значение свойства будет определено только после первого пакетного чтения, при условии, что в опциях задан флажок ReceiveTotalRecordsCount. Иначе свойство будет равно значению RecordCount..

8.19.19.3 Методы интерфейса

AddColumn

Метод добавляет колонку, значение которой должно быть получено в результате выполнения запроса. Дополнительно может быть указано преобразование, которое должно быть применено к содержимому колонки. Если преобразовывать содержимое колонки не требуется, то в качестве преобразования следует указать **null**.

Синтаксис:

```
void AddColumn (NodeColumn column, INodeColumnTransform transform);
```


Параметры:

column

Колонка, значение которой должно быть получено в результате выполнения запроса.

transform

Интерфейс для преобразования значений ячеек данной колонки. Если указать **null**, преобразование не требуется.

Execute

Метод выполняет запрос на чтение значений колонок для указанных дочерних элементов, либо на чтение порции дочерних элементов текущего элемента пространства навигации. Позиция для чтения определяется закладкой (bookmark). Если закладка равна **null**, то будет прочитана первая порция, иначе будет прочитана порция с позиции, указанной в закладке.

Синтаксис:

```
void Execute (NodeIDCollection nodeIDs);
```

```
void Execute (object bookmark, Int32 count);
```

Параметры:

nodeIDs

Коллекция идентификаторов дочерних элементов, содержимое колонок которых требуется получить из источника данных.

bookmark

Закладка, указывающая позицию для чтения. Если закладка равна **null**, то будет прочитана первая порция данных.

count

Количество записей, которое требуется получить в очередном пакете данных. В качестве значения можно использовать одну из констант класса QueryConsts.

GetRecordNodeID

Метод возвращает идентификатор полученного из источника данных дочернего элемента по его порядковому номеру.

Синтаксис:

```
INodeID GetRecordNodeID (Int32 index);
```

Параметры:

index

Порядковый номер требуемого дочернего элемента.

Результат:

Идентификатор полученного из источника данных дочернего элемента.

GetRecordValues

Метод возвращает значения колонок дочернего элемента по его порядковому номеру..

Синтаксис:

```
object[] GetRecordValues (Int32 index);
```

Параметры:

index

Порядковый номер требуемого дочернего элемента.

Результат:

Значения колонок дочернего элемента по его порядковому номеру.

8.19.20 Интерфейс IUpdateAnalyser

Интерфейс INodeItems, который позволяет управлять дочерними элементами пространства навигации, предоставляет механизм для обновления дочерних элементов по каким-то правилам. Для этих целей используются интерфейсы IUpdateAnalyser и IUpdatePlan.

8.19.20.1 Пространство имён

Intermech.Navigator.Interfaces

Интерфейс анализатора, позволяющий выполнить обработать интерфейс плана действий.

8.19.20.2 Методы интерфейса

Preprocess

Метод вызывается перед анализом списка элементов для выполнения предварительных действий.

Синтаксис:

```
void Preprocess (IUpdatePlan plan);
```

Параметры:

plan

Интерфейс плана по обновлению элементов пространства навигации.

Process

Метод вызывается для анализа каждого элемента из списка.

Синтаксис:

```
void Process (INodeID nodeID, IUpdatePlan plan);
```

Параметры:

nodeID

Идентификатор очередного элемента пространства навигации.

plan

Интерфейс плана по обновлению элементов пространства навигации.

Postprocess

Метод вызывается после анализа списка элементов для выполнения окончательных действий.

Синтаксис:

```
void Postprocess (IUpdatePlan plan);
```

Параметры:

plan

Интерфейс плана по обновлению элементов пространства навигации.

8.19.21 Интерфейс IUpdatePlan

Данный интерфейс позволяет выполнять какие-либо действия над очередным обновляемым дочерним элементом пространства навигации.

8.19.21.1 Пространство имён

Intermech.Navigator.Interfaces

Данный интерфейс используется для обновляемых дочерних элементов пространства навигации.

8.19.21.2 Методы интерфейса

Append

Метод вызывается, когда требуется добавить дочерний элемент пространства навигации в текущий список.

Синтаксис:

```
void Append (INodeID partialNodeID);
```

Параметры:

partialNodeID

Частично заполненный идентификатор элемента пространства навигации, который требуется добавить в список.

Remove

Метод вызывается, когда требуется удалить текущий дочерний элемент пространства навигации из списка.

Синтаксис:

```
void Remove ();
```

Replace

Метод вызывается, когда требуется заменить текущий дочерний элемент пространства навигации на указанный элемент.

Синтаксис:

```
void Replace (INodeID replacementNodeID);
```

Параметры:

replacementNodeID

Новый идентификатор элемента пространства навигации, которым требуется заменить текущий дочерний элемент.

Update

Метод вызывается, когда требуется обновить текущий дочерний элемент пространства навигации.

Синтаксис:

```
void Update ();
```

8.20 Работа с экспертной системой

Экспертная система (ЭС) позволяет проводить расчеты атрибутов, пересчеты атрибутов, зависящих от определенного атрибута, а также генерировать документы. Все эти задачи требуют предварительной подготовки различных объектов ЭС, что описано в гл. 4.4. Руководства администратора.

Все задачи экспертной системы выполняются на сервере. Пользовательский интерфейс экспертной системы позволяет запускать любую задачу ЭС с клиента и получать результаты. Он доступен только при наличии модуля экспертной системы (сборка **Intermech.Expert.User**).

Основным понятием при работе с экспертной системой является **задача ЭС**. Сервер экспертной системы может параллельно работать с несколькими задачами, как с одного и того же клиента, так и с различных. Задача ЭС хранит все рассчитанные и назначенные клиентскими приложениями значения атрибутов, которые в контексте этой задачи имеют приоритет над реальными значениями атрибутов, хранящимися в базе. Таким образом, различные задачи ЭС не мешают друг другу работать.

8.20.1 Перечислитель ExpertResult

Код результата, возвращаемый экспертной системой. Позволяет определить, выполнена ли задача, и если не выполнена, то почему.

8.20.1.1 Пространство имен

Intermech.Interfaces.Expert

8.20.1.2 Значения перечислителя:

Unknown

Результат неизвестен. Вероятнее всего, произошла какая-то ошибка, не имеющая специального значения перечислителя.

OK

Операция выполнена успешно.

WrongTaskId

Неверный идентификатор задачи. Обычно означает, что задача не была инициализирована или уже закрыта.

TaskBusy

Задача в данный момент занята. Следует дождаться завершения выполнения одной операции в контексте данной задачи перед началом другой операции.

NoContext

Контекст отсутствует. Большинство операций ЭС требуют наличия контекста – одного или нескольких объектов, для которых проводится операция.

NoSuitableObjects

Нет объектов ЭС, удовлетворяющих условиям. Выдается, если в правиле расчета атрибута не нашлось ни одного объекта ЭС, условия которого позволяли бы использовать его для расчета.

NoCondParms

Неизвестны параметры, необходимые для расчета условия. Список недостающих параметров можно получить функцией `GetMissingParms`.

NoCalcParms

Неизвестны параметры, необходимые для расчета. Список недостающих параметров можно получить функцией `GetMissingParms`.

ObjectNotFound

Объект не найден.

RuleNotFound

Правило расчета атрибута или правило поиска объектов не найдено.

AmbiguousValue

Результат расчета неоднозначен – существует несколько одинаково возможных вариантов.

CircularReference

Циклическое обращение объектов ЭС друг к другу.

EmptyTableCell

Результирующее значение представляет пустую клетку таблицы.

8.20.2 Перечислитель ExpertTraceFlags

Набор битовых флажков, определяющих объем передаваемой сервером информации трассировки.

8.20.2.1 Пространство имен

Intermech.Interfaces.Expert

8.20.2.2 Значения перечислителя:

None = 0

Не установлено флажков информации трассировки.

ShowExpertObjects = 1

Показывать информацию о загрузке объектов экспертной системы. Полезно для определения проблем, возникающих при загрузке правил расчета, формул, таблиц и прочих объектов ЭС.

ShowOtherObjects = 2

Показывать информацию о загрузке прочих объектов.

ShowObjConds = 4

Показывать информацию о проверке условий для объектов.

TraceAttribSearch = 8

Трассировать обработку поиска атрибутов.

TraceTables = 16

Трассировать обработку таблиц.

TraceScripts = 32

Трассировать обработку скриптов.

ShowContext = 64

Показывать объекты контекста.

ShowAttrChanges = 128

Показывать изменение атрибутов.

ShowObjResults=256

Показывать результат выборки объектов.

ShowScriptConds=512

Показывать проверку условий в операторах скриптов.

ShowFillDocs=1024

Показывать заполнение полей документов.

TraceObjectSearch = 2048

Трассировать поиск объектов.

ShowSettings = 4096

Показывать результаты присваивания.

8.20.3 Класс CalcAttrPair

Этот служебный класс предназначен для хранения параметров запроса, передаваемого экспертной системе при расчете параметров.

8.20.3.1 Пространство имен

Intermech.Interfaces.Expert

Класс для хранения параметров запроса расчета параметров. Тип объекта и тип атрибута образуют пару, для которой в экспертной системе задаются правила расчета атрибута.

8.20.3.2 Поля

Int64 objID

Идентификатор объекта, для которого рассчитывается параметр.

Int32 attrTypeID

Идентификатор рассчитываемого атрибута.

Int32 objTypeID

Идентификатор типа объекта.

8.20.3.3 Конструкторы

CalcAttrPair(Int64 objID, Int32 attrTypeID);

```
CalcAttrPair( Int64 objID, Int32 objTypeID, Int32 attrTypeID );
```

Параметры:

objID

Идентификатор объекта.

objTypeID

Идентификатор типа объекта.

attrTypeID

Идентификатор типа атрибута.

8.20.4 Класс CalculatedAttr

Класс предназначен для хранения параметров запроса для расчета параметров и результата расчета.

8.20.4.1 Пространство имен

Intermech.Interfaces.Expert

Класс для хранения значения параметров в контексте задачи ЭС.

8.20.4.2 Поля

CalcAttrPair ca_pair

Экземпляр класса CalcAttrPair, задающий параметры рассчитанного атрибута.

object Value

Значение атрибута.

bool Temporary

Является ли атрибут временным (если является, он не будет записан в базу).

8.20.4.3 Конструкторы

```
CalculatedAttr( Int64 objID, Int32 attrTypeID, object Val );
```

```
CalculatedAttr( CalcAttrPair ca_pair, object Val );
```

Параметры:

objID

Идентификатор объекта.

attrTypeID

Идентификатор типа атрибута.

ca_pair

Класс, задающий параметры атрибута.

Val

Значение атрибута.

8.20.5 Интерфейс IExpertTask

Данный интерфейс является основным средством доступа к функциям экспертной системы. Он инкапсулирует обращение к одной задаче ЭС. Получать интерфейс следует через интерфейс IExpertUser.

8.20.5.1 Пространство имен

Intermech.Interfaces.Expert

Данный интерфейс предоставляет доступ ко всем функциям экспертной системы.

8.20.5.2 Методы интерфейса

Calculate

Рассчитать значение атрибута в соответствии с правилами расчета (см. п. 4.4.1.2 Руководства администратора).

Методы, не имеющие параметра Value, после завершения расчета вызывают событие EndCalculate.

Синтаксис:

```
ExpertResult Calculate( Int32 attrTypeID, Int64 objId, out object Value );
```

```
ExpertResult Calculate( Int32 objTypeID, Int32 attrTypeID, Int64 objId, out object Value );
```

```
void Calculate( Int32 attrTypeID, Int64 objId );
```

```
void Calculate( Int32 objTypeID, Int32 attrTypeID, Int64 objId );
```

Параметры:

objTypeID

Тип объекта (первая половина пары).

attrTypeID

Тип атрибута (вторая половина пары).

objId

Идентификатор объекта (контекст).

Value

Возвращаемый результат.

GetMissingParms

Получить список недостающих параметров после расчета атрибута.

Синтаксис:

```
List<CalcAttrPair> GetMissingParms();
```

Результат:

Список атрибутов, которые экспертная система не смогла рассчитать.

GenerateDocument

Сгенерировать документ экспертной системы. Последний метод после завершения генерации вызывает событие OnEndGenerate.

Синтаксис:

```
ExpertResult GenerateDocument(Int64 docScriptID, Int64[] context, out object doc);
```

```
ExpertResult GenerateDocument(Int64 docScriptID, Int64[] context, Int64 docObjId);
```

```
void GenerateDocument(Int64 docScriptID, Int64[] context);
```

Параметры:

docScriptID

Идентификатор скрипта генерации документов.

context

Набор идентификаторов объектов, для которых генерируется документ.

doc

Сгенерированный документ (приводить к типу Intermech.Document.Model.ImDocument)

docObjId

Идентификатор объекта, в файл которого будет записан сгенерированный документ.

CalcFormula

Рассчитать результат формулы.

Синтаксис:

```
ExpertResult CalcFormula(object tf, Int64 objId, out object Value);
```

```
ExpertResult CalcFormula(Int64 formId, Int64 objId, out object Value);
```

Параметры:

tf

Формула экспертной системы (Intermech.Expert.TempFormula).

objId

Идентификатор объекта контекста.

Value

Результат расчета формулы.

formId

Идентификатор объекта формулы экспертной системы.

GetParmValue

Получить значение атрибута в контексте задачи.

Синтаксис:

```
object GetParmValue( Int64 objID, Int32 attrTypeID );
```

Параметры:

objID

Идентификатор объекта, для которого получаем значение.

attrTypeID

Идентификатор типа атрибута.

Возвращаемое значение:

Значение атрибута в контексте задачи или null, если значение не найдено.

SetParmValue

Установить значение атрибута в контексте задачи.

Синтаксис:

```
void SetParmValue( Int64 objID, Int32 attrTypeID, object Value );
```

Параметры:

objID

Идентификатор объекта, для которого устанавливается значение.

attrTypeID

Идентификатор устанавливаемого атрибута.

Value

Присваиваемое значение.

GetCalcParms

Получить значения всех атрибутов в контексте задачи.

Синтаксис:

```
Hashtable GetCalcParms();
```

Возвращаемое значение:

Таблица значений атрибутов в контексте задачи. Ключи таблицы имеют тип CalcAttrPair, значения – тип CalculatedAttr.

SetCalcParms

Установить значения всех атрибутов в контексте задачи.

Синтаксис:

```
void SetCalcParms( Hashtable parms );
```

Параметры:

parms

Таблица значений атрибутов в контексте задачи. Ключи таблицы имеют тип CalcAttrPair, значения – тип CalculatedAttr.

ApplyCalcParms

Записать значения атрибутов (кроме временных) в базу. Если список записываемых атрибутов не указан, записываются все возможные атрибуты.

Синтаксис:

```
void ApplyCalcParms();
```

```
void ApplyCalcParms(List<CalculatedAttr> list);
```

Параметры:

list

Список атрибутов, которые требуется записать в базу.

Recalculate

Пересчитать все атрибуты, которые зависят от указанного атрибута. При этом экспертная система использует скрипт пересчета для данного атрибута.

Синтаксис:

```
void Recalculate(Int32 attrTypeID, Int64 objId);
```

Параметры:

attrTypeID

Тип измененного атрибута.

objId

Идентификатор объекта, у которого был изменен атрибут.

Событие EndCalculate

Событие, возникающее при завершении расчета атрибута.

Синтаксис:

```
event EndCalculateEventHandler EndCalculate;
```

Делегат:

```
public delegate void EndCalculateEventHandler( object sender, EndCalculateEventArgs e );
```

Событие TimerTraceInfo

Событие, срабатывающее по таймеру и возвращающее информацию трассировки.

Синтаксис:

```
event GetTraceInfoEventHandler TimerTraceInfo;
```

Делегат:

```
public delegate void GetTraceInfoEventHandler( object sender, GetTraceInfoEventArgs e );  
TraceFlags
```

Прочитать или установить флажки, определяющие информацию трассировки, генерируемую сервером.

Синтаксис:

```
ExpertTraceFlags TraceFlags { get; set; }
```

InfoInterval

Прочитать или изменить интервал получения информации трассировки от сервера (в миллисекундах).

Синтаксис:

```
int InfoInterval { get; set; }
```

GetTraceinfo

Получить с сервера информацию трассировки, сгенерированную на данный момент.

Синтаксис:

```
XmlDocument GetTraceInfo();
```

Возвращаемое значение:

Информация трассировки на данный момент, в формате XML.

ShowTraceDialog

Показать диалоговое окно с информацией трассировки на данный момент.

Синтаксис:

```
void ShowTraceDialog();
```

8.20.6 Интерфейс IExpertUser

Данный интерфейс предназначен для получения интерфейса IExpertTask, а также выполняет другие вспомогательные функции.

8.20.6.1 Пространство имен

Intermech.Interfaces.Expert

8.20.6.2 Методы интерфейса

GetExpertTask

Получить интерфейс IExpertTask.

Синтаксис:

```
IExpertTask GetExpertTask();
```

Возвращаемое значение:

Ссылка на интерфейс IExpertTask.

ShowTraceWindow

Возвращает пользовательскую настройку, будет ли выдана информация трассировки.

Синтаксис:

```
bool ShowTraceWindow{ get; }
```

Возвращаемое значение:

Истина, если пользователь включил выдачу информации трассировки.

9 Структура файлов портфеля

9.1 Папки портфеля

Папка	Содержимое
\	Файлы портфеля
\Blob	Двоичные данные (тип ftBlob, ftFile)
\Memo	Текстовые данные (тип ftMemo)
\ShortBlob	Короткие двоичные данные (тип ftShortBlob)

9.2 Файлы портфеля

Имя файла	Содержание
BriefcaseConfig.xml	Состояние портфеля
ExportContent.xml	Список указанных для экспорта данных
Metadata.xml	Метаданные
MetadataExportList.xml	Список экспортированных метаданных
MetadataSecurity.xml	Настройки безопасности на метаданные
ObjAttributes.xml	Дополнительные атрибуты объектов
Objects.xml	Объекты
ObjLcSteps.xml	Нахождение объектов на шагах ЖЦ
ObjSecurity.xml	Настройки безопасности на версии объектов
Relations.xml	Связи
RelAttributes.xml	Дополнительные атрибуты связей
Contexts.xml	Содержимое контекстов

В файлах портфеля идентификаторы могут быть указаны в виде глобальных идентификаторов (GUID) и в виде локальных идентификаторов (внутренних) экспортируемой базы.

Глобальные идентификаторы указываются в портфеле для однозначной идентификации метаданных, объектов и связей при их верификации во время процесса импорта. Локальные идентификаторы указываются для сохранения взаимосвязей в пределах портфеля, а также для ускорения поисковых процессов при обработке портфеля на этапе импорта. Если явно не указано является идентификатор глобальным или локальным, подразумевается, что идентификатор локальный.

Даты и время в портфеле указываются в виде: уууу-мм-ддThh:mm:ss, где уууу – год, мм-месяц, dd-день, T-разделитель между датой и временем, hh-часы, mm-минуты, ss-секунды, время приведено в UTC (по Гринвичу).

Информация по типам данных различных полей, записываемых в xml файлы, находится в файлах схем xml с именами <имя xml файла портфеля>.xsd.

9.2.1 BriefcaseConfig.xml

Файл, описывающий состояние портфеля.

Пример.

```
<configuration>
  <briefcase name="Briefcase" comment="" version="0" exportdate="12/12/2006 15:03:15"
lastsystemupdate="12/12/2006 14:39:00" closed="True" />
</configuration>
```

, где

- name – наименование портфеля, оно же имя папки, в которой находится портфель
- comment – комментарий
- version – зарезервировано

- exportdate – дата экспорта, время в UTC
- lastsystemupdate – время последнего изменения метаданных в экспортируемой базе данных
- closed = "True" – если портфель был закрыт после успешного окончания экспорта данных.

Даты и время в файле состояния портфеля указаны в виде строки в инвариантной культуре (CultureInfo.InvariantCulture), время приведено в UTC (по Гринвичу).

9.2.2 ExportContent.xml

Файл, содержащий список данных, указанных для экспорта во время создания портфеля.

Пример.

```
<EXPORTCONTENTDATASET>
<EXPORTATTRIBUTE>
  <F_CATEGORY_ID>1</F_CATEGORY_ID>
  <F_OBJECT_ID>103fa428-f4be-4c0a-ac08-19c801d2fb9e</F_OBJECT_ID>
</EXPORTATTRIBUTE>
...
</EXPORTCONTENTDATASET>
```

Между тэгами <EXPORTATTRIBUTE> </EXPORTATTRIBUTE> находится информация по указанным для экспорта метаданным:

- F_CATEGORY_ID – категория (см. п. [9.3.1](#))
- F_OBJECT_ID – глобальный идентификатор.

9.2.3 Metadata.xml

Файл, содержащий описание всех метаданных экспортируемой базы.

Является сохраненным в виде xml содержимым Dataset, полученным следующим образом:

```
IServerBriefcase iServerBriefcase = (session as IUserSession).GetBriefcase() as IServerBriefcase;
iServerBriefcase.StartExport();
Dataset metadataDataSet = iServerBriefcase.GetDataset( new string[] { "SYSTEM" } );
metadataDataSet.DataSetName = "METADATASET";
```

При сохранении в xml полученного metadataDataSet файл имеет следующий вид:

```
<METADATASET>
<!-- запись таблицы групп атрибутов -->
<IMS_ATTR_GROUPS> ... </IMS_ATTR_GROUPS>
<!-- запись таблицы групп атрибутов -->
<IMS_ATTR_GROUPS> ... </IMS_ATTR_GROUPS>
...
<!-- запись таблицы атрибутов -->
<IMS_ATTRIBUTES> ... </IMS_ATTRIBUTES>
<!-- запись таблицы атрибутов -->
<IMS_ATTRIBUTES> ... </IMS_ATTRIBUTES>
...
<!-- запись таблицы принадлежности атрибутов к группам атрибутов-->
<IMS_ATTR_IN_GROUPS> ... </IMS_ATTR_IN_GROUPS>
...
<!-- запись таблицы языковых вариантов -->
<IMS_LANGUAGES>...</IMS_LANGUAGES>
...
<!-- записи по остальным таблицам метаданных-->
...
</METADATASET> <!-- dataset -->
```

По назначению и составу полей соответствующих метаданных в xml файле см. раздел [3](#).

9.2.4 MetadataExportList.xml

Файл, содержащий список метаданных из общего состава метаданных (см. п. [9.2.2](#)), задействованных в процессе экспорта базы.

Пример.

```
<METADATAEXPORTLIST>
<!-- запись-->
<METADATA>
<category>7</category> <id>10018</id> <external>fd6aa87c-919a-42c5-ba53-1098c75d4e62</external>
</METADATA>
<!-- запись-->
<METADATA>
  <category>8</category> <id>2</id> <external>cad00013-306c-11d8-b4e9-00304f19f545</external>
</METADATA>
...
</METADATAEXPORTLIST>
```

между тэгами <METADATA> </METADATA> находится по три элемента для каждого метаданного:

- category – категория метаданного (см. п. [9.3.1](#))
- id – локальный идентификатор метаданного в экспортируемой базе
- external – глобальный идентификатор метаданного.

9.2.5 Objects.xml

Файлы, содержащие информацию по объектам и версиям объектов.

Пример.

```
<OBJECTSDATASET>
<OBJECT>
  <F_OBJECT_ID>160832</F_OBJECT_ID>
  <F_OBJECTGUID>0232dc7c-4774-47f0-9f73-f990df52bc88</F_OBJECTGUID>
  <F_ID>160833</F_ID>
  <F_IDGUID>e7cd9c57-e3e6-46d6-bf5c-e521ddb8afe2</F_IDGUID>
  <F_LC_STEP>10028</F_LC_STEP>
  <F_CHKOUT_BY></F_CHKOUT_BY>
  <F_CHKOUTGUID></F_CHKOUTGUID>
  <F_VERSION_ID>0</F_VERSION_ID>
  <F_OBJECT_VER_TYPE>0</F_OBJECT_VER_TYPE>
  <F_OBJECT_TYPE>1002</F_OBJECT_TYPE>
  <F_OWNER_ID>58477</F_OWNER_ID>
  <F_OWNERGUID>9b560d97-16f7-4ecb-9bde-4cb393e4d83b</F_OWNERGUID>
  <F_MODIFY_DATE>11/22/2006 13:26:06</F_MODIFY_DATE>
  <F_LEVEL_ID>2</F_LEVEL_ID>
  <F_OBJ_CREATE>10/04/2006 06:05:02</F_OBJ_CREATE>
  <CAPTION>Объект </CAPTION>
</OBJECT>
</OBJECTSDATASET>
```

Между тэгами <OBJECT> и </OBJECT> находятся значения обязательных атрибутов объекта (см. также п. [5.1.12](#)):

- F_OBJECT_ID – локальный идентификатор версии объекта
- F_OBJECTGUID – глобальный идентификатор версии объекта
- F_ID – локальный идентификатор объекта
- F_IDGUID – глобальный уникальный идентификатор объекта
- F_LC_STEP - идентификатор этапа жизненного цикла
- F_CHKOUT_BY – локальный идентификатор пользователя, взявшего версию объекта на редактирование (может не указываться, если не взят на изменение)
- F_CHKOUTGUID – глобальный идентификатор пользователя, взявшего версию объекта на редактирование (может не указываться, если не взят на изменение)
- F_VERSION_ID - порядковый номер версии объекта

- F_OBJECT_VER_TYPE - признак версии/экземпляра/актуальной версии
- F_OBJECT_TYPE - идентификатор типа объекта
- F_OWNER_ID - идентификатор объекта-владельца
- F_OWNERGUID – уникальный идентификатор объекта-владельца
- F_MODIFY_DATE - дата последней модификации объекта
- F_LEVEL_ID - идентификатор уровня продвижения
- F_OBJ_CREATE - дата создания версии объекта
- CAPTION – заголовок объекта

9.2.6 Relations.xml

Файлы, содержащие информацию по связям.

Пример.

```
<RELATIONSDATASET>
  <RELATION>
    <F_PRJLINK_ID>160836</F_PRJLINK_ID>
    <F_PROJ_ID>0232dc7c-4774-47f0-9f73-f990df52bc88</F_PROJ_ID>
    <F_PART_ID>4b89e26e-aabf-4cdc-95c3-85bc7d2042fe</F_PART_ID>
    <F_RELATION_TYPE>1014</F_RELATION_TYPE>
    <F_CREATE_DATE>10/04/2006 06:03:29</F_CREATE_DATE>
  </RELATION>
  ...
</RELATIONSDATASET>
```

Между тэгами <RELATION> и </RELATION> находятся значения обязательных атрибутов связи.

- F_PRJLINK_ID – локальный идентификатор связи
- F_PRJ_GUID – глобальный идентификатор связи
- F_PROJ_ID – глобальный идентификатор версии родительского объекта
- F_PART_ID – глобальный идентификатор дочернего объекта (не версии)
- F_RELATION_TYPE – идентификатор типа связи
- F_CREATE_DATE – дата создания связи
- F_DELETE_DATE – дата удаления связи

9.2.7 ObjAttributes.xml, RelAttributes.xml

Файлы, содержащие информацию по дополнительным атрибутам применительно к объектам и связям.

Пример (для ObjAttributes.xml):

```
<OBJATTRIBUTESDATASET>
  <ATTRIBUTE>
    <F_ATTRIBUTE_ID>1212</F_ATTRIBUTE_ID>
    <F_OBJECT_ID>160834</F_OBJECT_ID>
    <F_INLIST_ID>0</F_INLIST_ID>
    <F_DATE_VALUE>10/04/2006 07:03:28</F_DATE_VALUE>
  </ATTRIBUTE>
  <ATTRIBUTE>
    <F_ATTRIBUTE_ID>14</F_ATTRIBUTE_ID>
    <F_OBJECT_ID>92655</F_OBJECT_ID>
    <F_INLIST_ID>0</F_INLIST_ID>
    <F_DOUBLE_VALUE>1E-06</F_DOUBLE_VALUE>
  </ATTRIBUTE>
  ...
</OBJATTRIBUTESDATASET>
```

Пример (для RelAttributes.xml)

```

<RELATTRIBUTESDATASET>
<ATTRIBUTE>
  <F_ATTRIBUTE_ID>1212</F_ATTRIBUTE_ID>
  <F_PRJLINK_ID>1251</F_OBJECT_ID>
  <F_INLIST_ID>0</F_INLIST_ID>
  <F_DATE_VALUE>10/04/2006 07:03:28</F_DATE_VALUE>
</ATTRIBUTE>
<ATTRIBUTE>
  <F_ATTRIBUTE_ID>14</F_ATTRIBUTE_ID>
  <F_PRJLINK_ID>1273</F_OBJECT_ID>
  <F_INLIST_ID>0</F_INLIST_ID>
  <F_DOUBLE_VALUE>1E-06</F_DOUBLE_VALUE>
</ATTRIBUTE>
</RELATTRIBUTESDATASET>

```

Между тэгами <ATTRIBUTE> и </ATTRIBUTE> находятся поля информации об атрибуте, принадлежности объекту/связи, а также, собственно, поля значений атрибутов.

Состав полей идентификации значений атрибутов постоянен и представлен следующими полями:

- F_ATTRIBUTE_ID – идентификатор типа атрибута
- F_OBJECT_ID – идентификатор версии объекта (только для атрибутов на объект)
- F_PRJLINK_ID – идентификатор связи (только для атрибутов на связь)
- F_INLIST_ID – порядковый номер значения атрибута (начинается с 0)

Состав полей значений атрибутов динамичен и заполняется в зависимости от типа атрибута:

Поле	ftString	ftInteger	ftDouble	ftDateTime	ftShortBlob	ftFile	ftExternalLink
F_INTEGER_VALUE	-	число	-	-	идентификатор блока коротких двоичных данных	идентификатор файла	-
F_INTEGERGUID	резерв	резерв	резерв	резерв	резерв	резерв	резерв
F_DOUBLE_VALUE	-	-	вещ. число	-	-	идентификатор файлового шкафа	-
F_DOUBLEGUID	резерв	резерв	резерв	резерв	резерв	резерв	резерв
F_STRING_VALUE	строка	-	-	-	комментарий	имя файла	-
F_DATE_VALUE	-	-	-	дата, время (по Гринвичу)	дата последней модиф.	дата последней модификации файла	-
F_FILESIZE	-	-	-	-	реальный размер блока коротких двоичных данных	реальный размер файла	-
F_ARC_METHOD	-	-	-	-	0 – не упакован 1 – zip	0 – не упакован 1 – zip	-

F_NOTE	-	-	-	-	-	комментарий к файлу	-
F_PATH2FILE	резерв	резерв	резерв	резерв	резерв	резерв	резерв

	ftObjectLink	ftPassword	ftMemo	ftBlob	ftBoolean	ftMeasured	ftAutoInc	ftGuid
F_INTEGER_VALUE	идентификатор версии объекта	-	идентификатор блока текста	идентификатор блока двоичных данных	0 false 1 true	идентификатор базовой единицы измерения	число	-
F_INTEGERGUID	резерв	резерв	резерв	резерв	резерв	резерв	резерв	резерв
F_DOUBLE_VALUE	-	-	-	идентификатор файлового шкафа	-	значение в базовой единице измерения	-	-
F_DOUBLEGUID	резерв	резерв	резерв	резерв	резерв	резерв	резерв	резерв
F_STRING_VALUE	заголовок объекта	хэш пароля	текст, первые 450 символов	комментарий к блоку двоичных данных	-	строковое значение в единицах измерения, выбранных пользователем	-	строковое представление guid
F_DATE_VALUE	-	дата назначения пароля	дата последней модификации	дата последней модиф.	-	-	-	-
F_FILESIZE	-	-	размер текста	реальный размер блока	-	-	-	-
F_ARC_METHOD	-	-	0	0 – не упакован 1 – zip	-	-	-	-
F_NOTE	-	-	-	комментарий к блоку (совпадает с F_STRIN	-	-	-	-

F_PATH2FILE	резерв	резерв	резерв	G_VALUE) резерв	резерв	резерв	резерв	резерв

Атрибуты типов ftMemo, ftFile, ftShortBlob и ftBlob помимо указанных в xml значений имеют также блоки данных, которые сохраняются в подпапках соответствующих базовых папок (см. п. 9.1).

Путь к файлу относительно базовых папок формируется следующим образом: подпапка с именем из двух последних шестнадцатеричных цифр от идентификатора объекта/связи, подпапка с именем из двух последних шестнадцатеричных цифр от идентификатора атрибута, подпапка с именем из двух последних шестнадцатеричных цифр от идентификатора блока двоичных данных/текста/файла.

Имя конечного файла представлено дополненным слева нулями до 8 символов шестнадцатеричным представлением идентификатора блока двоичных данных/текста/файла. Расширение файла: .mem для текстового атрибута, .blb для атрибута файла или двоичных данных.

Таким образом, файлы находятся на третьем уровне относительно базовых папок, количество подпапок в каждой папке ограничено 256 папками.

9.2.8 ObjLcSteps.xml

Файл, содержащий информацию по переходу версий объектов на этапы жизненного цикла.

Пример.

```
<LCSTEPSDATASET>
  <LCSTEP>
    <F_OBJECT_ID>157603</F_OBJECT_ID>
    <F_LC_STEP>10000</F_LC_STEP>
    <F_START_DATE>11/20/2006 08:54:35</F_START_DATE>
  </LCSTEP>
```

...

```
</LCSTEPSDATASET>
```

между тэгами <LCSTEP> </LCSTEP> находятся следующие данные:

- F_OBJECT_ID – идентификатор версии объекта
- F_LC_STEP – идентификатор шага жизненного цикла
- F_START_DATE – дата и время перемещения версии объекта на шаг жизненного цикла (переведенное в UTC)

9.2.9 MetadataSecurity.xml, ObjSecurity.xml

Файлы содержат настройки безопасности применительно к метаданным и версиям объектов соответственно.

Пример (для MetadataSecurity.xml)

```
<METADATASECURITYDATASET>
  <SECURITY>
    <F_CATEGORY_ID>1033</F_CATEGORY_ID>
    <F_CATEGORY_TYPE>4</F_CATEGORY_TYPE>
    <F_RIGHT_ID>18</F_RIGHT_ID>
    <F_USER_ID>cad00393-306c-11d8-b4e9-00304f19f545</F_USER_ID>
    <F_RIGHT_TYPE>1</F_RIGHT_TYPE>
    <F_OWNER_ID>dafdd127-141c-4184-8d89-d8184d668c48</F_OWNER_ID>
    <F_BEGIN_DATE>12/14/2006 00:00:00</F_BEGIN_DATE>
    <F_END_DATE>12/21/2006 00:00:00</F_END_DATE>
  </SECURITY>
```

...

```
</METADATASECURITYDATASET>
```

Пример (для ObjSecurity.xml)

```
<OBJSECURITYDATASET>
  <SECURITY>
    <F_CATEGORY_ID>191181</F_CATEGORY_ID>
    <F_CATEGORY_TYPE>1</F_CATEGORY_TYPE>
```

```

<F_RIGHT_ID>18</F_RIGHT_ID>
<F_USER_ID>a0e39b41-f082-44ab-9d1d-936e7395c6cf</F_USER_ID>
<F_RIGHT_TYPE>1</F_RIGHT_TYPE>
<F_OWNER_ID>1d6ff53a-690a-4698-9c52-3d409ae8a535</F_OWNER_ID>
</SECURITY>

```

...

```
</OBJSECURITYDATASET>
```

между тэгами <SECURITY> </SECURITY> находятся следующие данные (список полей аналогичен получаемому при помощи функции IDBSecurity.GetAccessList()):

- F_CATEGORY_TYPE – идентификатор категории (см. п. [9.3.1](#)).
- F_CATEGORY_ID – идентификатор в рамках категории
- F_USER_ID – идентификатор пользователя, которому назначается право доступа
- F_OWNER_ID – идентификатор пользователя, назначившего право доступа
- F_RIGHT_ID – идентификатор вида действия (см. п. [9.3.3](#))
- F_RIGHT_TYPE – категория действия (см. п. [9.3.2](#))
- F_BEGIN_DATE – дата начала действия права доступа
- F_END_DATE – дата окончания действия права доступа

9.2.10 Contexts.xml

Файл содержит информацию по содержимому контекстов.

Пример (для Contexts.xml)

```

<CONTEXTSDATASET>
  <CONTEXT id="1270096" modification_id="0" content="1270098;1270100" />
</CONTEXTSDATASET>

```

Тэг <CONTEXT> содержит информацию о содержимом контекстов:

- id – идентификатор контекста;
- modification_id – номер взаимосвязанного контекста;
- content – содержимое контекста: список идентификаторов версий объектов через «;».

9.3 Системные константы

9.3.1 Категории

Идентификатор категории	Наименование категории
0	Неизвестная
1	Версия объекта
2	Объект
3	Атрибут
4	Тип объекта
5	Связь
6	Тип связи
7	Шаг жизненного цикла
8	Уровень продвижения
9	Языковой вариант
10	Журнал событий
11	Предметная область

12	Группа атрибутов
13	Зарезервировано
14	Системная
15	Файлы
16	Схема жизненного цикла
17	Документ

9.3.2 Категории действий

Идентификатор	Наименование
0	Не определено
1	Читать
2	Изменять
3	Администрировать

9.3.3 Вид действия

Идентификатор	Значение	Наименование
Create	0	Создание
CreateVersion	1	Создание версии
Edit	2	Редактирование
EditProperties	3	Изменение свойств
Delete	4	Удаление
Remove	5	Перемещение
Read	6	Чтение
Write	7	Запись
View	8	Просмотр
Open	9	Открытие
Execute	10	Выполнение
AddLink	11	Создание связи
EditLink	12	Изменение связи
DeleteLink	13	Удаление связи
List	14	Получение списка
Compute	15	Вычисление
Print	16	Печать
Copy	17	Копирование
GetAccess	18	Получение прав доступа
SetAccess	19	Изменение прав доступа
Login	20	Вход в систему
GetLinks	21	Получение связей
Export	22	Экспорт
Import	23	Импорт
Send	24	Отправка

Идентификатор	Значение	Наименование
NextLCStep	25	Изменение шага ЖЦ
Purge	26	Уничтожение
Clear	27	Очистка
Cancel	28	Отмена
CheckOut	29	Взятие на изменение
CheckIn	30	Завершение изменений
TakeOwnership	31	Изменение владельца
Save	32	Сохранение изменений
Load	33	Загрузка данных
Any	34	Действие не определено
SaveToDisk	35	Копирование документа на диск
DocRegistry	36	Регистрация документа в ОТД
IncludeInComposition	43	Включение в состав
ExcludeFromComposition	44	Исключение из состава
ChangeBaseVersion	45	Изменение базовой версии
Restore	46	Восстановление итерации
ShowHistory	47	Просмотр истории значений атрибутов
ChangeAccessLevel	48	Изменение уровня доступа объекта
ShowNonApplicabilityImbaseRecords	51	Отображение записей с отключенной применяемостью
ShowNonVisibleColumnImbaseRecords	52	Отображение записей с отключенной видимостью колонки
ShowNonVisibleRowImbaseRecords	53	Отображение записей с отключенной видимостью строки
CreateFolderOrRecordInCatalog	54	Создание папки (записи каталога) в каталоге
CreateTableLinkInCatalog	55	Создание таблицы (ярлыка таблицы) в каталоге
Use	56	Использовать
ShowNonUseImbaseRecords	57	Отображение запрещенных к использованию записей
EditTableStructureAndProperties	58	Редактирование структуры и свойств таблицы
EditTableData	59	Редактирование данных в таблице
ManageCatalogIndexes	60	Управление индексами каталога
AddNewRows	61	Добавление новых записей
SignDocument	62	Подписание документа
wfLaunchProcess	1001	Инициировать процессы по шаблону
wfEditProcess	1002	Редактировать порожденный процесс
wfAdminProcess	1003	Администрировать порожденный процесс

10 Формы редактирования и элементы ввода и отображения информации

10.1.1 Интерфейс IFormDesignerControl

Данный интерфейс должны наследовать все элементы управления, используемые в редакторе форм, которые могут являться контейнерами для других элементов управления. А также могут наследовать элементы управления, которые должны получать от формы уведомления о завершении загрузки данных и о деактивации формы.

10.1.1.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.1.2 События интерфейса

FormDeactivate

При деактивации формы рассылает данное уведомление дочерним элементам.

Синтаксис:

```
event EventHandler FormDeactivate;
```

LoadDataCompleted

Уведомляет дочерние элементы о том, что процесс загрузки данных на форму завершен.

Синтаксис:

```
event EventHandler LoadDataCompleted;
```

10.1.1.3 Свойства интерфейса

CanContainsChildren

Возможность элемента управления выступать в качестве контейнера для других элементов управления.

Синтаксис:

```
bool CanContainsChildren { get; }
```

10.1.2 Класс FormDesignerFormLinksProviderType

Данные провайдера, предоставляющего информацию об элементах, которым назначена форма редактирования. Необходимы для IFormDesignerFormLinksManager, чтобы зарегистрировать провайдер.

10.1.2.1 Пространство имен

Intermech.Interfaces

10.1.2.2 Конструктор

FormDesignerFormLinksProviderType(string name, Guid guid, Type type);

Параметры:

name

Наименование провайдера.

guid

Глобальный идентификатор провайдера.

type

Тип провайдера.

10.1.2.3 Поля класса

ProviderGuid

Глобальный идентификатор провайдера.

Синтаксис:

```
Guid ProviderGuid
```

ProviderName

Наименование провайдера.

Синтаксис:

```
string ProviderName
```

ProviderType

Тип данных провайдера.

Синтаксис:

Type ProviderType

10.1.3 Интерфейс IFormDesignerFormLinksManager

Сервис, с помощью которого можно работать с зарегистрированными в системе провайдерами, предоставляющими информацию об элементах, которым назначена форма редактирования. Получить интерфейс IFormDesignerFormLinksManager можно у статического класса ServicesManager.

10.1.3.1 Пространство имен

Intermech.Interfaces

10.1.3.2 Методы интерфейса

RegisterProvider

Зарегистрировать провайдер.

Синтаксис:

```
void RegisterProvider(FormDesignerFormLinksProviderType providerType)
```

Параметры:

providerType

Данные о регистрируемом провайдере.

UnregisterProvider

Удалить провайдер.

Синтаксис:

```
void UnregisterProvider(Guid providerGuid)
```

Параметры:

providerGuid

Глобальный идентификатор провайдера.

GetProvider

Получить данные о провайдере по его глобальному идентификатору.

Синтаксис:

```
FormDesignerFormLinksProviderType GetProvider(Guid providerGuid)
```

Параметры:

providerGuid

Глобальный идентификатор провайдера.

Возвращаемое значение:

Данные о провайдере.

10.1.4 Класс FormLink

При реализации провайдера, предоставляющего информацию об элементах, которым назначена форма редактирования, необходимо реализовать класс, который будет наследоваться от FormLink. Хранит информацию об атрибутах элемента, которые можно связать с контролами редактирования.

10.1.4.1 Пространство имен

Intermech.Interfaces

10.1.4.2 Поле класса

ProviderGuid

Глобальный идентификатор провайдера.

10.1.4.3 Виртуальное свойство класса

Attributes

Список атрибутов, которые можно получить у элемента и связать их с контролами редактирования.

Синтаксис:

```
virtual List<Int32> Attributes { get; }
```

10.1.5 Интерфейс IFormDesignerFormLinksProvider

Интерфейс, который необходимо реализовать провайдеру, для предоставления информации об элементах, которым назначена форма редактирования.

10.1.5.1 Пространство имен

Intermech.Interfaces

10.1.5.2 Свойства интерфейса

Loaded

Результат загрузки информации о типах объектов и связей, а также объектов и связей, которым назначена форма.

Синтаксис:

```
bool Loaded { get; }
```

ProviderGuid

Глобальный идентификатор провайдера

Синтаксис:

```
Guid ProviderGuid { get; }
```

ProviderName

Наименование провайдера

Синтаксис:

```
string ProviderName { get; }
```

RootNode

При реализации интерфейса, необходимо сформировать узел дерева **TreeView**. Он содержит элементы, которым назначена форма редактирования.

Синтаксис:

```
object RootNode { get; }
```

Возвращаемое значение:

Корневой узел, с элементами-подузлами, которым назначена форма редактирования.

FormLinks

Список объектов класса **FormLink**, содержащих информацию об атрибутах элементов.

Синтаксис:

```
List<FormLink> FormLinks { get; }
```

10.1.5.3 Методы интерфейса

Load

Метод для добавления элементов, которые связаны с формой редактирования, в корневой узел **RootNode**.

Синтаксис:

```
void Load(Int64 formID)
```

Параметры:

formID

Идентификатор формы редактирования.

Add

Добавление элемента в корневой узел **RootNode**.

Синтаксис:

```
void Add()
```

Delete

Удаление элемента.

Синтаксис:

```
void Delete(object node)
```

Параметры:

node

Выбранный элемент узла **RootNode**.

Clear

Очистить корневой узел RootNode.

Синтаксис:

```
void Clear()
```

Commit

Сохранение измененной информации.

Синтаксис:

```
void Commit()
```

Rollback

Отмена изменений информации.

Синтаксис:

```
void Rollback()
```

10.1.6 Интерфейс IFormDesignerActionParams

Интерфейс дополнительных параметров.

10.1.6.1 Пространство имен

Intermech.Interfaces

10.1.6.2 Свойство интерфейса

Component

Компонент.

Синтаксис:

```
object Component { get; set; }
```

10.1.7 Класс FormDesignerAction

Класс для регистрации дополнительных обработчиков событий кнопки или формы.

10.1.7.1 Пространство имен

Intermech.Interfaces

10.1.7.2 Свойства класса

ActionGuid

Глобальный идентификатор обработчика.

Синтаксис:

```
Guid ActionGuid { get; set; }
```

ActionName

Наименование обработчика.

Синтаксис:

```
string ActionName { get; set; }
```

ActionParams

Дополнительные параметры.

Синтаксис:

```
IFormDesignerActionParams ActionParams { get; set; }
```

10.1.7.3 Конструкторы класса

```
FormDesignerAction(Guid actionGuid, string actionName)
```

```
FormDesignerAction(Guid actionGuid, string actionName, IFormDesignerActionParams actionParams)
```

10.1.8 Интерфейс IFormDesignerActionHandler

Интерфейс описания обработчиков событий для кнопки.

10.1.8.1 Пространство имен

Intermech.Interfaces

10.1.8.2 Методы интерфейса

ButtonEnabled

Определение доступности кнопки.

Синтаксис:

```
bool ButtonEnabled(object button, object form)
```

Параметры:

button

Кнопка, для которой происходит определение активности.

form

Форма редактирования.

ButtonPressed

Нажатие на кнопку.

Синтаксис:

```
void ButtonPressed(object button, object form)
```

Параметры:

button

Кнопка, для которой происходит определение активности.

form

Форма редактирования.

10.1.9 Интерфейс IFormDesignerActionManager

Сервис, с помощью которого можно работать с дополнительными обработчиками событий кнопок. Получить интерфейс IFormDesignerActionManager можно у статического класса ServicesManager.

10.1.9.1 Пространство имен

Intermech.Interfaces

10.1.9.2 Методы интерфейса

RegisterAction

Регистрация обработчика.

Синтаксис:

```
void RegisterAction(Guid actionGuid, string actionName, IFormDesignerActionHandler handler)
```

Параметры:

actionGuid

Глобальный идентификатор обработчика.

actionName

Наименование обработчика.

handler

Обработчик событий для кнопки.

RegisterAction

Синтаксис:

```
void RegisterAction(FormDesignerAction action, IFormDesignerActionHandler handler)
```

UnregisterAction

Удаление обработчика.

Синтаксис:

```
void UnregisterAction(Guid actionGuid)
```

Параметры:

actionGuid

Глобальный идентификатор обработчика.

GetInfo

Информация об обработчике.

Синтаксис:

```
FormDesignerAction GetInfo(Guid actionGuid);
```

Параметры:

actionGuid

Глобальный идентификатор обработчика.

GetAction

Получение обработчика событий для кнопки.

Синтаксис:

```
IFormDesignerActionHandler GetAction(object actionInfo)
```

Параметры:

actionInfo

Объект для получения обработчика событий. Может передаваться либо информация об обработчике (экземпляр класса FormDesignerAction), либо глобальный идентификатор обработчика.

10.1.10 Интерфейс IFormDesignerEventHandlerBase

Базовый интерфейс описания обработчиков событий для формы.

10.1.11 Интерфейс IFormDesignerFormEventsHandler

Интерфейс описания обработчиков событий для формы. Наследуется от интерфейса IFormDesignerEventHandlerBase.

10.1.11.1 Пространство имен

Intermech.Interfaces

10.1.11.2 Методы интерфейса

DataLoaded

Окончание загрузки данных. Срабатывает после загрузки всех данных. Также возникает после отмены изменений.

Синтаксис:

```
void DataLoaded(object form)
```

Параметры:

form

Форма редактирования.

Deactivate

Закрытие формы. Срабатывает после закрытия формы и после Deactivate'a вьюшки.

Синтаксис:

```
void DataLoaded(object form)
```

Параметры:

form

Форма редактирования.

Modified

Модификация данных. Срабатывает после модификации данных (при первой модификации после загрузки)

Синтаксис:

```
void Modified(object form)
```

Параметры:

form

Форма редактирования.

10.1.12 Интерфейс IAttributeChangingEventHandler

Интерфейс описания обработчиков событий для изменения атрибутов на форме редактирования. Наследуется от интерфейса IFormDesignerEventHandlerBase.

10.1.12.1 Пространство имен

Intermech.Interfaces.Client

10.1.12.2 Методы интерфейса

AttributeChanging

Изменение значений атрибутов. Может возникать как при редактировании значения атрибута в элементе управления, так и при изменении значений атрибутов другими модулями (например экспертной системой).

Синтаксис:

```
void AttributeChanging(AttributeChangingEventArgs args)
```


Параметры:

args

Данные формы (см. описание класса `AttributeChangingEventArgs`).

10.1.13 Класс `AttributeChangingEventArgs`

Класс с данными формы, который передается наблюдателю события, возникающего при изменении значений атрибутов на форме редактирования.

10.1.13.1 Пространство имен

`Intermech.Interfaces.Client`

10.1.13.2 Методы класс

`FormID`

Идентификатор формы редактирования.

Синтаксис:

```
public Int64 FormID { get; set; }
```

`ObjectID`

Идентификатор основного объекта/связи.

Синтаксис:

```
public Int64 ObjectID { get; set; }
```

`ObjectTypeID`

Тип основного объекта/связи.

Синтаксис:

```
public Int32 ObjectTypeID { get; set; }
```

`OldObjectAttributes`

Старые (до изменения) значения атрибутов основного объекта/связи.

Синтаксис:

```
public IEnumerable<AttributeValues> OldObjectAttributes { get; set; }
```

`NewObjectAttributes`

Новые значения атрибутов основного объекта/связи. Сюда подписчики записывают значения атрибутов, после их изменения. Измененные значения вернутся в форму и будут отображены в контролах.

Синтаксис:

```
public IEnumerable<AttributeValues> NewObjectAttributes { get; set; }
```

`RelationID`

Идентификатор дополнительной связи.

Синтаксис:

```
public Int64 RelationID { get; set; }
```

`RelationTypeID`

Тип дополнительной связи.

Синтаксис:

```
public Int32 RelationTypeID { get; set; }
```

`OldRelationAttributes`

Старые (до изменения) значения атрибутов дополнительной связи.

Синтаксис:

```
public IEnumerable<AttributeValues> OldRelationAttributes { get; set; }
```

`NewRelationAttributes`

Новые значения атрибутов дополнительной связи. Сюда подписчики записывают значения атрибутов, после их изменения. Измененные значения вернутся в форму и будут отображены в контролах.

Синтаксис:

```
public IEnumerable<AttributeValues> NewRelationAttributes { get; set; }
```

10.1.14 Интерфейс IFormDesignerEventManager

Сервис, с помощью которого можно работать с дополнительными обработчиками событий формы редактирования. Получить интерфейс IFormDesignerEventManager можно у статического класса ServicesManager.

10.1.14.1 Пространство имен

Intermech.Interfaces

10.1.14.2 Методы интерфейса

GetEvent

Получение обработчика события.

Синтаксис:

```
IFormDesignerEventHandlerBase GetEvent(Guid eventGuid)
```

Параметры:

eventGuid

Глобальный идентификатор события

GetEvents

Получение списка событий по его типу.

Синтаксис:

```
Dictionary<Guid, FormDesignerAction> GetEvents(Type eventType)
```

Параметры:

eventType

Тип обработчика событий.

GetInfo

Получение описания события.

Синтаксис:

```
FormDesignerAction GetInfo(Guid eventGuid)
```

Параметры:

eventGuid

Глобальный идентификатор события

RegisterEvent

Регистрация события.

Синтаксис:

```
void RegisterEvent(Type eventType, Guid eventGuid, string eventName, IFormDesignerEventHandlerBase eventHandler)
```

Параметры:

eventType

Тип обработчика событий.

eventGuid

Глобальный идентификатор события

eventType

Наименование события.

eventHandler

Обработчик события.

RegisterEvent

Регистрация события.

Синтаксис:

```
void RegisterEvent(Type eventType, FormDesignerAction action, IFormDesignerEventHandlerBase eventHandler)
```

Параметры:

eventType

Тип обработчика событий.

action

Описание обработчика.

eventHandler

Обработчик события.

UnregisterEvent

Удаление обработчика событий.

Синтаксис:

```
void UnregisterEvent(Type eventType, Guid eventGuid)
```

Параметры:

eventType

Тип обработчика событий.

eventGuid

Глобальный идентификатор события

10.1.15 Класс AttributeInfo

Класс, который хранит информацию об атрибуте.

10.1.15.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.15.2 Конструктор

AttributeInfo()

AttributeInfo(Guid attributeGuid, Guid typeGuid)

Параметры:

attributeGuid

Глобальный идентификатор типа атрибута.

typeGuid

Глобальный идентификатор типа объектов/связи, которому принадлежит атрибут.

10.1.15.3 Свойства класса

AttributeGuid

Глобальный идентификатор типа атрибута.

Синтаксис:

```
Guid AttributeGuid { get; set; }
```

TypeGuid

Глобальный идентификатор типа объектов/связи, которому принадлежит атрибут.

Синтаксис:

```
Guid TypeGuid { get; set; }
```

10.1.16 Перечисление RefreshMode

Перечисление, значения которого используются для задания режима обновления данных на форме редактирования.

10.1.16.1 Пространство имен

Intermech.Client.Core.FormDesigner.Navigator

10.1.16.2 Значения

Default

По умолчанию.

Forced

Принудительно.

10.1.17 Класс DesForm

Форма ввода и редактирования информации. Реализует интерфейс IFormDesignerControl.

10.1.17.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.17.2 Конструктор

DesForm()

10.1.17.3 Свойства класса

AdditionalValues

Возвращает список данных атрибутов, для объекта или связи, которые были изменены, но которые не связаны с каким-либо элементом управления, расположенным на форме.

Синтаксис:

```
Dictionary<Int64, List<AttributeValues>> AdditionalValues { get; }
```

Возвращаемое значение:

Словарь данных атрибутов, для объекта и/или связи, которые были изменены, но которые не связаны с каким-либо элементом управления, расположенным на форме.

Ключ - идентификатор объекта/связи

Значение – список данных атрибутов

AttrButtons

Возвращает список кнопок, расположенных на форме редактирования.

Синтаксис:

```
List<AttrButton> AttrButtons { get; }
```

Возвращаемое значение:

Список кнопок, расположенных на форме редактирования.

FormID

Идентификатор формы редактирования.

Синтаксис:

```
Int64 FormID { get; set; }
```

Возвращаемое значение:

Идентификатор формы редактирования.

Info

Информация об основном рассматриваемом объекте/связи. Состоит из идентификатора объекта/связи и информации о принадлежности элемента к объекту/связи.

Синтаксис:

```
IElementInfo Info { get; set; }
```

Возвращаемое значение:

Информация об основном рассматриваемом объекте/связи.

Установить значение свойства Info необходимо до начала загрузки данных на форму.

InfoReadOnly

Возможность редактирования основного объекта/связи.

Синтаксис:

```
bool InfoReadOnly { get; }
```

Возвращаемое значение:

Возможность редактирования элемента.

RelationInfo

Информация о связи, если объект рассматривается в составе. Если рассматривается обособленно связь, то данные для этой связи можно получить через свойство **Info**.

Синтаксис:

```
IElementInfo RelationInfo { get; set; }
```

Links

Список провайдеров, предоставляющих информацию об элементах, которым назначена форма редактирования

Синтаксис:

```
FormLinks Links { get; set; }
```

FormDesignerEvents

Список обработчиков событий для формы.

Синтаксис:

```
FormDesignerAction[] FormDesignerEvents { get; set; }
```

HelpPartLabel

Метка раздела в файле справки, назначенного форме редактирования.

Синтаксис:

```
string HelpPartLabel { get; set; }
```

HelpPathToFile

Путь к файлу справки, назначенного форме редактирования.

Синтаксис:

```
string HelpPathToFile { get; set; }
```

Modified

Наличие измененных значений атрибутов.

Синтаксис:

```
bool Modified { get; }
```

10.1.17.4 Методы класса

GetInformation

Список элементов редактирования расположенных на форме.

Синтаксис:

```
Dictionary<AttributeInfo, IAttributeEditor> GetInformation()
```

Возвращаемое значение:

Список элементов редактирования (исключая кнопки). В качестве ключа используется информация об атрибуте, связанном с элементом. В качестве значения используется сам элемент редактирования.

GetPinInformation

Список элементов редактирования расположенных на форме, которым назначены атрибуты.

Синтаксис:

```
Dictionary<AttributeInfo, IAttributeEditor> GetPinInformation(Int64 ID)
```

Параметры:

ID

Идентификатор объекта/связи, для которых необходимо получить список контролов.

Возвращаемое значение:

Список элементов редактирования (исключая кнопки), которым назначены атрибуты. В качестве ключа используется информация об атрибуте, связанном с элементом. В качестве значения используется сам элемент редактирования.

LoadAttributes

Загрузка данных объекта или связи.

Синтаксис:

```
void LoadAttributes()
```

```
void LoadAttributes(RefreshMode mode)
```

Параметры:

mode

Режим обновления данных.

SetFormDeactivate

Деактивация формы. Необходимо вызывать, когда происходит деактивация формы. В частности вызывается вьюшкой при ее деактивации. Рассылает сообщение о деактивации формы всем дополнительным обработчикам событий.

Синтаксис:

```
void SetFormDeactivate()
```

UpdateSlaveAttribute

При изменении в элементах управления значений для мастера атрибутов возникает необходимость обновить значения связанных с ним атрибутов.

Синтаксис:

```
void UpdateSlaveAttribute(IElementInfo elementInfo, AttributeValuesList attrValueList)
```

Параметры:

`elementInfo`

Информация об объекте или связи.

`attrValueList`

Список объектов, которые содержат данные редактируемого атрибута.

SaveAttributes

Сохранение измененных значений атрибутов.

Синтаксис:

```
void SaveAttributes()
```

```
bool SaveAttributes(bool blankMode)
```

Параметры:

`blankMode`

Признак того что форма работает с заготовкой объекта.

Возвращаемое значение:

Результат сохранения.

ValidateBeforeSave

Выполнение у элемента управления дополнительных действий перед сохранением значения.

Синтаксис:

```
void ValidateBeforeSave()
```

10.1.18 Интерфейс `IBaseDesForm`

Форма редактирования, которой принадлежит элемент управления.

10.1.18.1 *Пространство имен*

Intermech.Client.Core.FormDesigner.Controls

10.1.18.2 *Свойство интерфейса*

DesForm

Форма редактирования.

Синтаксис:

```
DesForm DesForm { set; }
```

10.1.19 Интерфейс `IAttributeEditorModified`

Интерфейс отслеживания и обработки изменений в элементе управления.

10.1.19.1 *Пространство имен*

Intermech.Client.Core.FormDesigner.Controls

10.1.19.2 *Свойство интерфейса*

Modified

Наличие изменения данных.

Синтаксис:

```
bool Modified { get;set; }
```

10.1.19.3 *Событие интерфейса*

ModifiedEvent

Событие на изменение данных в контроле

Синтаксис:

```
event EventHandler ModifiedEvent
```

10.1.20 Интерфейс `IAttributeEditor`

Интерфейс для работы с данными атрибута, назначенного элементу управления. Наследуется от интерфейсов `IBaseDesForm` и `IAttributeEditorModified`.

10.1.20.1 *Пространство имен*

Intermech.Client.Core.FormDesigner.Controls

10.1.20.2 Свойства интерфейса

AttributeInfo

Информация об атрибуте.

Синтаксис:

```
AttributeInfo AttributeInfo { get;set; }
```

AttributeInfo

Возможность добавления атрибута объекту/связи если таковой отсутствует. При условии, что типу объектов/связи разрешено добавить соответствующий атрибут.

Синтаксис:

```
bool CanAddAttribute { get;set; }
```

Values

Информация об атрибуте.

Синтаксис:

```
AttributeValues Values { get;set; }
```

10.1.20.3 Метод интерфейса

SetPossibleValues

Метод предназначен для случая, когда атрибут может содержать список возможных значений.

Синтаксис:

```
void SetPossibleValues(DataTable data)
```

Параметры:

data

Таблица возможных значений. Необходимо получать у объекта IDBAttributeType с помощью метода GetPossibleValues().

10.1.21 Перечисление AttributeDestinationPoint

Принадлежность атрибута элементу. Атрибут для контролов редактирования может браться либо у основного объекта/связи, либо у дополнительной связи, если объект рассматривается в составе.

10.1.21.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.21.2 Значения

Default

Атрибута основного объекта/связи

Relation

Атрибута дополнительной связи

10.1.22 Интерфейс IParent4Control

Интерфейс для установки родителя контрола.

10.1.22.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.22.2 Свойства интерфейса

ParentInfo

Содержит информацию об элементе, которому принадлежит контрол.

Синтаксис:

```
IElementInfo ParentInfo { get;set; }
```

ParentPoint

Принадлежность атрибута элементу.

Синтаксис:

```
AttributeDestinationPoint ParentPoint { get;set; }
```

10.1.23 Интерфейс IValidateBeforeSave

Интерфейс наследуют контролы, у которых перед сохранением данных необходимо выполнить некоторые действия, например, необходимо проверять правильность заполнения.

10.1.23.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.23.2 Методы интерфейса

Validate

Выполнение дополнительных действий перед сохранением данных.

Синтаксис:

```
void Validate()
```

10.1.24 Интерфейс IImageFromLibrary

Интерфейс для объектов, у которых есть возможность загружать изображение из библиотеки изображений.

10.1.24.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.24.2 Свойство интерфейса

ImageFromLibrary

Глобальный идентификатор объекта "библиотечное изображение".

Синтаксис:

```
Guid ImageFromLibrary { get; set; }
```

ImageFromLibraryName

Наименование объекта "библиотечное изображение".

Синтаксис:

```
string ImageFromLibraryName { get; }
```

ImageFromLibraryID

Идентификатор объекта "библиотечное изображение".

Синтаксис:

```
Int64 ImageFromLibraryID { get; }
```

ImageFromLibraryImage

Изображение.

Синтаксис:

```
Image ImageFromLibraryImage { get; }
```

10.1.25 Класс IMLabel

Элемент управления создан для возможности отображения в нем изображений, хранящихся в библиотеке изображений IPS Development Kit. Реализует интерфейс IImageFromLibrary

10.1.25.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.25.2 Конструктор

IMLabel()

10.1.26 Класс IMPictureBox

Элемент управления создан для возможности отображения в нем изображений, хранящихся в библиотеке изображений IPS Development Kit. Реализует интерфейс IImageFromLibrary

10.1.26.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.26.2 Конструктор

IMPictureBox()

10.1.27 Класс AttrCheckedListBox

Элемент редактирования, предназначенный для выбора нескольких значений из списка возможных значений. Реализует интерфейсы IAttributeEditor и IParent4Control.

10.1.27.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.27.2 Конструктор

[AttrCheckedListBox\(\)](#)

10.1.28 Класс AttrComboBox

Элемент редактирования, предназначенный для выбора одного значения из списка возможных значений. Реализует интерфейсы IAttributeEditor и IParent4Control.

10.1.28.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.28.2 Конструктор

[AttrComboBox](#)

10.1.28.3 Свойства класса

[EnableWithoutAttribute](#)

Для данного элемента управления можно разрешить доступ, даже если он не связан с каким-либо атрибутом.

Синтаксис:

```
bool EnableWithoutAttribute { get; set; }
```

10.1.29 Класс AttrDateEdit

Элемент редактирования, предназначенный для работы с датой и временем. Реализует интерфейсы IAttributeEditor и IParent4Control.

10.1.29.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.29.2 Конструктор

[AttrDateEdit\(\)](#)

10.1.30 Интерфейс IIMControlEnabled

Интерфейс, с помощью которого можно контролировать доступность дочерних элементов управления, при «недоступности» родительского. Для этого, при необходимости выставить родительскому элементу управления свойство Enabled в значение false, нужно воспользоваться свойством интерфейса EnabledCtrl и сделать недоступными только те дочерние элементы, которые необходимо.

10.1.30.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.30.2 Свойство интерфейса

[EnabledCtrl](#)

Реализация недоступности дочерних элементов.

Синтаксис:

```
bool EnabledCtrl { get;set; }
```

10.1.31 Класс IMGroupBox

Элемент редактирования, предназначенный для группировки контролов. Реализует интерфейсы IFormDesignerControl, IImageFromLibrary и IIMControlEnabled.

10.1.31.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.31.2 Конструктор

IMGroupBox()

10.1.32 Класс IMPanel

Элемент редактирования, предназначенный для группировки контролов. Реализует интерфейсы IFormDesignerControl, IImageFromLibrary и IIMControlEnabled.

10.1.32.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.32.2 Конструктор

IMPanel ()

10.1.33 Класс IMTabControl

Элемент редактирования, предназначенный для группировки контролов. Реализует интерфейсы IFormDesignerControl и IIMControlEnabled.

10.1.33.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.33.2 Конструктор

IMTabControl()

10.1.34 Класс AttrButton

Элемент управления – кнопка. По нажатию выполняет действия, определенные пользователем. Реализует интерфейс IBaseDesForm.

10.1.34.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.34.2 Конструктор

AttrButton()

10.1.34.3 Свойства класса

FormDesignerAction

Пользовательский обработчик события для кнопки.

Синтаксис:

```
FormDesignerAction FormDesignerAction { get; set; }
```

FormDesignerActionParams

Дополнительные параметры.

Синтаксис:

```
IFormDesignerActionParams FormDesignerActionParams { get; set; }
```

10.1.35 Класс AttrsControl

Базовый класс для элементов редактирования, которые связаны с атрибутами. Реализует интерфейсы IAttributeEditor, IParent4Control и IIMControlEnabled.

10.1.35.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.35.2 Конструктор

AttrsControl()

10.1.35.3 Свойство класса

SetError

Установить для контрола сообщение об ошибке (например, что атрибут не может содержать пустых значений).

Синтаксис:

```
string SetError { get; set; }
```

10.1.36 Класс AttrCheckBox

Элемент управления для работы с логическими значениями. Возможна работа с экспертной системой. Наследник класса AttrsControl. Реализует интерфейс IImageFromLibrary.

10.1.36.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.36.2 Конструктор

[AttrCheckBox\(\)](#)

10.1.36.3 Свойство класса

[UseInExpertSystem](#)

Возможность использовать значение в экспертной системе.

Синтаксис:

```
bool UseInExpertSystem { get; set; }
```

10.1.37 Класс AttrLabel

Элемент управления для просмотра текстового значения атрибута. Возможна работа с экспертной системой. Наследник класса AttrsControl. Реализует интерфейс IImageFromLibrary.

10.1.37.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.37.2 Конструктор

[AttrLabel\(\)](#)

10.1.37.3 Свойство класса

[UseInExpertSystem](#)

Возможность использовать значение в экспертной системе.

Синтаксис:

```
bool UseInExpertSystem { get; set; }
```

10.1.38 Класс AttrTextEdit

Элемент управления для изменения текстовых значений с возможностью просмотра истории. Возможна работа с экспертной системой. Наследник класса AttrsControl.

10.1.38.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.38.2 Конструктор

[AttrTextEdit\(\)](#)

10.1.38.3 Свойства класса

[UseInExpertSystem](#)

Возможность использовать значение в экспертной системе.

Синтаксис:

```
bool UseInExpertSystem { get; set; }
```

[HasDescriptor](#)

Наличие у атрибута описателя IAttributePropertyDescriptor.

Синтаксис:

```
bool HasDescriptor { get; }
```

10.1.38.4 Событие класса

[TxtKeyDown](#)

Нажатие клавиши клавиатуры при нахождении курсора в поле редактирования.

Синтаксис:

event KeyEventHandler TxtKeyDown

10.1.39 Класс AttrMaskedTextEdit

Элемент управления для изменения текстовых значений по маске. С возможностью просмотра истории. Возможна работа с экспертной системой. Наследник класса AttrsControl.

10.1.39.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.39.2 Конструктор

[AttrMaskedTextEdit\(\)](#)

10.1.39.3 Свойства класса

[UseInExpertSystem](#)

Возможность использовать значение в экспертной системе.

Синтаксис:

```
bool UseInExpertSystem { get; set; }
```

[HasDescriptor](#)

Наличие у атрибута описателя IAttributePropertyDescriptor.

Синтаксис:

```
bool HasDescriptor { get; }
```

10.1.39.4 Событие класса

[TxtKeyDown](#)

Нажатие клавиши клавиатуры при нахождении курсора в поле редактирования.

Синтаксис:

```
event KeyEventHandler TxtKeyDown
```

10.1.40 Класс AttrMemoEdit

Элемент управления для изменения многострочных текстовых значений. Возможна работа с экспертной системой. Наследник класса AttrsControl.

10.1.40.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.40.2 Конструктор

[AttrMemoEdit\(\)](#)

10.1.40.3 Свойства класса

[UseInExpertSystem](#)

Возможность использовать значение в экспертной системе.

Синтаксис:

```
bool UseInExpertSystem { get; set; }
```

10.1.41 Класс AttrPassword

Элемент управления для работы с паролями. Наследник класса AttrsControl.

10.1.41.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.41.2 Конструктор

[AttrPassword\(\)](#)

10.1.42 Класс AttrTextBtn

Элемент управления для работы со ссылочными атрибутами. Наследник класса AttrsControl.

10.1.42.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.42.2 Конструктор

AttrTextBtn()

10.1.42.3 Свойство класса

EnabledCtrl

Переопределено для того, чтобы можно было сделать доступной кнопку «Карточка объекта», если само значение редактировать нельзя и контрол недоступен.

Синтаксис:

```
override bool EnabledCtrl { get; set; }
```

10.1.42.4 Событие класса

TxtKeyDown

Нажатие клавиши клавиатуры при нахождении курсора в поле редактирования.

Синтаксис:

```
event KeyEventHandler TxtKeyDown
```

10.1.42.5 ValueChanged

Изменение значения.

Синтаксис:

```
event KeyEventHandler ValueChanged
```

10.1.43 Класс AttrTextBtnComp

Элемент управления для работы с атрибутами, которые ссылаются на объекты, созданные на основе справочников IMBASE. Наследник класса AttrsControl.

10.1.43.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.43.2 Конструктор

AttrTextBtnComp()

10.1.44 Класс AttrMeasuredEdit

Элемент управления для работы с атрибутами, выраженными в единицах измерения. Возможна работа с экспертной системой. Наследник класса AttrsControl.

10.1.44.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.44.2 Конструктор

AttrMeasuredEdit()

10.1.44.3 Свойства класса

UseInExpertSystem

Возможность использовать значение в экспертной системе.

Синтаксис:

```
bool UseInExpertSystem { get; set; }
```

Button

Кнопка.

Синтаксис:

```
Button Button { get; }
```

10.1.44.4 Событие класса

TxtKeyDown

Нажатие клавиши клавиатуры при нахождении курсора в поле редактирования.

Синтаксис:

```
event KeyEventHandler TxtKeyDown
```

10.1.45 Класс AttrListBoxBase

Базовый класс для списочных элементов управления, которые связаны с атрибутами, способными содержать несколько значений. Наследник класса AttrsControl.

10.1.45.1 Пространство имен

Intermech.Client.Core.FormDesigner.Controls

10.1.45.2 Конструктор

AttrListBoxBase ()

10.1.45.3 Свойство класса

HasDescriptor

Наличие у атрибута описателя IAttributePropertyDescriptor.

Синтаксис:

```
bool HasDescriptor { get; }
```

10.1.45.4 Событие класса

CreateItemForListBox

Создание нового элемента для списка. Наследник создает элемент, нужного ему класса.

Синтаксис:

```
protected event ListItemEventHandler CreateItemForListBox
```

GetItemForAttributeValues

Получение элемента. Наследник определяет, какое значение должен сохранить атрибут и возвращает это значение для записи в AttributeValues.

Синтаксис:

```
protected event ListItemEventHandler GetItemForAttributeValues
```

MenuItemClick

Выбор пункта контекстного меню.

Синтаксис:

```
protected event EventHandler MenuItemClick
```

10.1.45.5 Делегат класса

Синтаксис:

```
protected delegate object ListItemEventHandler(object item)
```

Параметры:

item

Значение из списка значений атрибута.

Возвращаемое значение:

Элемент.

10.1.45.6 Методы класса

CheckAccessibilityButtons

Проверка доступности кнопок и пунктов меню.

Синтаксис:

```
protected void CheckAccessibilityButtons()
```

CheckModified

Проверка изменения данных.

Синтаксис:

```
protected void CheckModified()
```

DeleteItem

Удаление элемента.

Синтаксис:

```
protected bool DeleteItem()
```

10.1.46 Класс AttrListBox

Элемент управления для работы со списком строковых значений (строковые типы, целочисленные, вещественные, глобальный идентификатор). Наследник класса AttrListBoxBase.

10.1.46.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.46.2 Конструктор

[AttrListBox\(\)](#)

10.1.47 Класс AttrListBox

Элемент управления для работы со списком ссылочных значений. Наследник класса AttrListBoxBtn.

10.1.47.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.47.2 Конструктор

[AttrListBoxBtn\(\)](#)

10.1.48 Класс AttrMeasuredListBox

Элемент управления для работы со списком значений, выраженных в единице измерения. Наследник класса AttrListBoxBtn.

10.1.48.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.48.2 Конструктор

[AttrMeasuredListBox\(\)](#)

10.1.49 Перечисление ListContext

Перечисление для правила подбора объектов элемента управления ObjectsList.

10.1.49.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.49.2 Значения

Objects

Отображаются все объекты, либо объекты указанного типа.

Composition

Отображается состав указанного объекта.

Applicability

Отображается применяемость указанного объекта.

10.1.50 Класс ObjectsList

Элемент управления для отображения списка объектов. Реализует интерфейс IFormDesignerControl.

10.1.50.1 Пространство имен

[Intermech.Client.Core.FormDesigner.Controls](#)

10.1.50.2 Конструктор

[ObjectsList\(\)](#)

10.1.50.3 Свойства класса

[ColumnCollection](#)

Набор определенных пользователем колонок.

Синтаксис:

NodeColumnCollection ColumnCollection { get; set; }

[ColumnsAliases](#)

Пользователь может сам задать наименование (псевдонимы) выбранных колонок.

Синтаксис:

```
Dictionary<Guid, string> ColumnsAliases { get; set; }
```

Возвращаемое значение:

Список колонок. В качестве ключа выступает глобальный идентификатор атрибута, в качестве значения введенный пользователем псевдоним.

DataSourceName

В качестве источника данных, может выступать другой элемент управления ObjectsList.

Синтаксис:

```
string DataSourceName { get; set; }
```

Возвращаемое значение:

Наименование контрола-источника данных.

List

В свойстве можно указать правило подбора объектов.

Для ListContext.Objects можно указать в свойстве ObjectsTypeID тип объектов, которые будут отображаться в элементе управления.

Для ListContext.Composition и ListContext.Applicability можно кроме типа объектов задать еще фильтр по типу связи.

Синтаксис:

```
ListContext List { get; set; }
```

ObjectsTypeID

Идентификатор типа объектов, по которому будут фильтроваться отображаемые объекты.

Синтаксис:

```
int ObjectsTypeID { get; set; }
```

ObjectTypesGuid

Глобальный идентификатор типа объектов, по которому будут фильтроваться отображаемые объекты.

Синтаксис:

```
Guid ObjectTypesGuid { get; set; }
```

RelationsTypeID

Идентификатор типа связи, по которому будут фильтроваться отображаемые объекты.

Синтаксис:

```
int RelationsTypeID { get; set; }
```

SelectionID

Идентификатор контекстной выборки, в соответствии с условиями которой, будет производиться подбор отображаемых объектов.

Синтаксис:

```
Int64 SelectionID { get; set; }
```

SelectedItems

Список выделенных, в элементе управления, объектов.

Синтаксис:

```
ICollection SelectedItems { get; }
```

SelectedItemsIDs

Список идентификаторов, выделенных в элементе управления объектов.

Синтаксис:

```
List<Int64> SelectedItemsIDs { get; }
```

ShowStatusBar

Отображение строки состояния.

Синтаксис:

```
bool ShowStatusBar { get; set; }
```

SourceCtrl

Контроль, который выступает в качестве источника данных.

Синтаксис:


```
ObjectsList SourceCtrl { get; set; }
```

UseColumnsAliases

Возможность использовать псевдонимы колонок вместо их наименований.

Синтаксис:

```
bool UseColumnsAliases { get; set; }
```

10.1.50.4 Событие класса

SelectedItemsChanged

Изменение выделенного объекта.

Синтаксис:

```
event EventHandler SelectedItemsChanged
```

10.2 Примеры

10.2.1 Создание пользовательского события для кнопки

1. Реализуем интерфейс `Intermech.Interfaces.IFormDesignerActionHandler`

```
public class TestButtonActionHandler : IFormDesignerActionHandler
{
    #region IFormDesignerActionHandler Members
    /// <summary>
    /// Проверка состояния кнопки Enabled/Disabled.
    /// </summary>
    /// <param name="button">Кнопка (AttrButton)</param>
    /// <param name="form">Форма (DesForm)</param>
    /// <returns>true - если кнопка должна быть Enabled</returns>
    public bool ButtonEnabled(object button, object form)
    {
        // TODO: проверяем условие, при котором кнопка должна быть доступна
        return true;
    }
    /// <summary>
    /// Событие на нажатие кнопки.
    /// </summary>
    /// <param name="button">Кнопка (AttrButton)</param>
    /// <param name="form">Форма (DesForm)</param>
    public void ButtonPressed(object button, object form)
    {
        // TODO: Событие, которое должно происходить при нажатии на кнопку
        MessageBox.Show("Hello World!!!", "Salutation", MessageBoxButtons.OK,
            MessageBoxIcon.Hand);
    }
    #endregion
}
```

2. Создаем статический класс **ActionsInfo** со статическим полем типа **FormDesignerAction** (для удобства)

```
public static class ActionsInfo
{
    // TODO: GUID новый, который будет идентифицировать данное событие
    public static FormDesignerAction TestButtonAction = new FormDesignerAction(new
        System.Guid("2FD62EA3-2312-488F-9BE1-E4A5966E52D9"), "Тестовое событие");
}
```

3. Регистрируем действие на загрузке плагина

```
IFormDesignerActionManager actionManager =
    ServicesManager.GetService(typeof(IFormDesignerActionManager)) as IFormDesignerActionManager;
```

```
if (actionManager != null)
{
    actionManager.RegisterAction(ActionsInfo.TestButtonAction, new TestButtonActionHandler());
}
```

11 Разработка сценариев на языке C#

В системе IPS Development Kit присутствует возможность написания пользовательских сценариев на языке C# для автоматизации каких-либо действий. Для этого существует несколько видов сценариев: сценарии маршрутизатора, планировщика задач, генерации документов и т.д.

11.1 Модель сценариев C#

В общем виде, сценарий C# представляет собой исходный текст кода на C#, который «на лету» компилируется и собирается в сборку .NET, которая в режиме выполнения подгружается сервером приложений либо клиентом. После чего из сборки создается экземпляр класса и вызывается метод.

Более подробно этот процесс описан ниже.

11.2 Компиляция сценариев

Сценарии компилируются с использованием технологии программных доменов AppDomain, а выполняются в специально подготовленном изолированном окружении внутри основного AppDomain приложения. Применение AppDomain для компиляции позволяет изолировать сценарии как друг от друга, так и от IPS Development Kit. Благодаря этому решены следующие задачи:

1. Обеспечивается потокобезопасное выполнение нескольких копий одного и того же компилятора;
2. Обеспечивается контроль за потреблением сервером приложений IPS Development Kit оперативной памяти в условиях длительной работы.

Подробнее об AppDomain можно прочитать на сайте Microsoft по [ссылке](#).

11.3 Выполнение сценариев

Выполнение сценария C# происходит по следующей схеме:

1. Компиляция кода сценария производится при первом обращении к нему. Если в дальнейшем код сценария изменяется, то IPS Development Kit автоматически выполняет его перекомпиляцию. Для компиляции используется вспомогательный домен из пула свободных доменов;
2. Скомпилированный код сценария и все необходимые сборки загружаются в основной домен приложения;
3. В основном домене приложения подготавливается изолированное окружение для выполнения сценария;
4. Создается объект класса Script;
5. У созданного объекта сценария заполняется свойство ScriptContext, а также любые другие свойства, типы значений которых соответствуют типам сервисов IPS API. Благодаря этим свойствам сценарий сможет обращаться к IPS API;
6. У созданного объекта сценария вызывается метод Execute(), в который передаются параметры из той подсистемы IPS Development Kit, для которой предназначен сценарий;
7. После завершения выполнения сценария производится очистка объекта сценария и изолированного окружения. Сначала очищается свойство ScriptContext и свойства-ссылки на сервисы IPS API, а затем освобождается ссылка на сам объект сценария.

11.4 Требования к коду сценариев C#

Для того чтобы сценарий C# мог быть выполнен, его код должен удовлетворять следующим требованиям:

1. В коде сценария должен присутствовать нестатический класс Script;

- 1.1. У класса Script должен быть конструктор по умолчанию (т.е. без параметров). Допускается отсутствие любых конструкторов у класса Script, так как в этом случае конструктор по умолчанию будет создан компилятором;
- 1.2. У класса Script должен быть нестатический метод Execute(). Список параметров метода не фиксирован и зависит от подсистемы, для которой предназначен сценарий;
- 1.3. У класса Script должно быть нестатическое свойство ScriptContext типа ICSSharpScriptContext (из пространства имен Intermech.Interfaces);
2. В коде сценария, помимо класса Script, могут присутствовать и другие типы;
3. У всех классов в коде сценария не должно быть статических полей данных, кроме тех, которые помечены модификатором readonly.

11.5 Проверка кода сценариев

В клиенте IPS Development Kit существует команда **Настройка/Утилиты администратора/Проверить код сценариев C#**, которая проверяет код всех сценариев C# и находит те сценарии, которые не соответствуют требованиям описанным в п. [11.4](#). После выполнения команды формируется HTML отчет, в котором подробно расписано, какие сценарии по какой причине требуют корректировки.

11.6 Обязательные условия корректного выполнения сценариев

При написании сценариев обязательно соблюдение следующих условий:

1. Объект сценария должен освобождаться сразу после выполнения, это накладывает ряд ограничений:
 - 1.1. запрещено использование фоновых потоков и всего, что использует фоновые потоки:
 - класс ThreadPool и классы, использующие его (Timer, Task и т.п.);
 - модификаторы Async и Await;
 - методы BeginInvoke и EndInvoke;
 - Parallel LINQ;
 - и т.п.
 - 1.2. запрещена подписка на события клиента/сервера – это не дает очистить объект сценария
 - 1.3. запрещено использование модификатора static – один и тот же сценарий может одновременно выполняться в нескольких потоках и использование static нарушает потокобезопасность выполняемого кода
2. Обработка ошибок:
 - 2.1 в сценариях требуется избегать StackOverflowException – т.к. данную ошибку невозможно перехватить блоком try/catch, что в свою очередь влечет падение основной программы (сервера приложений/клиента)
 - 2.2 в сценариях, выполняемых на стороне сервера, запрещено использовать сборку System.Windows.Forms – если сервер приложения служба, то использование элементов UI повлечет падение сервера приложений.
3. Работа с транзакциями
 - 3.1 в случае использования механизма транзакций в сценариях требуется обязательно использовать блок try/catch/finally, чтобы открытая транзакция в любом случае была либо завершена, либо отменена.

11.7 Отладка сценариев

Отлаживать сценарий можно при помощи Visual Studio. Для этого требуется в тексте сценария в метод Execute добавить следующие строки:

```
if (Debugger.IsAttached)
```

```
{
    Debugger.Break();
}
```

А в список используемых пространств имен добавить `using System.Diagnostics;`

Для запуска сценария на отладку, перед выполнением сценария требуется в Visual Studio выполнить команду `Debug – Attach to Process...` и в списке процессов выбрать сервер приложений либо клиент (в зависимости от того где будет выполняться сценарий). После чего запустить сценарий на выполнение. После чего в Visual Studio откроется текст сценария и можно будет пошагово выполнить код.

11.8 Рекомендации при написании сценариев:

1. При написании кода сценария использовать `MetaDataHelper`.

Как правило в сценариях требуется привязка к каким-либо метаданным (типам объектов, атрибутов, связей и т.п.). Записывать в текст сценария идентификатор метаданных (id) не рекомендуется, т.к. бывают ситуации, когда id измениться. Вместо этого рекомендуется использовать глобальный идентификатор guid метаданных (только по guid можно однозначно идентифицировать метаданные), а по нему при помощи `MetaDataHelper` получать id. Также рекомендуется всячески использовать `MetaDataHelper` для получения информации о метаданных вместо вызова соответствующих методов сессии `IUserSession` – это сократит использование обращений к БД, что в свою очередь позитивно скажется на производительности сервера приложений.

2. При получении состава/применимости/выборку объектов в перечень запрашиваемых данных выводить все требуемые данные, вместо `ObjectId`.

Например, требуется по составу получить список используемых материалов.

Некорректное решение, вначале получить перечень всех идентификаторов, входящих в состав. А потом при помощи `IUserSession.GetObject()` получать каждый входящий объект и у него запрашивать атрибут **Материал**.

Корректное решение, в параметры запроса сразу добавить колонку, в которой будет выведен атрибут **Материал**.

11.9 Примеры сценариев C#

11.9.1 Пример сценария Workflow.

```
using System;
using Intermech.Interfaces;
using Intermech.Interfaces.Workflow;

public class Script
{
    /// <summary>
    /// Свойство ScriptContext для обращения к IPS API.
    /// Значение свойства заполняется автоматически непосредственно перед вызовом метода Execute.
    /// </summary>
    public ICSHarpScriptContext ScriptContext
    {
        get;
        set;
    }

    /// <summary>
    /// Пример использования сервиса IPS API.
    /// Значение свойства заполняется автоматически непосредственно перед вызовом метода Execute.
    /// </summary>
    public IOutputView OViewService
    {
        get;
        set;
    }

    public void Execute(IActivity activity)
    {

```

```

        // вместо обращения к MetadataHelper следует использовать ScriptContext.MetadataHelper
        // например, получим название типа объектов для пользователя 'Системный администратор'
        var sysdbaUser = activity.Session.GetObject(sysdbaUserGuid, true);
        var sysdbaTypeName = ScriptContext.MetadataHelper.GetObjectTypeName(sysdbaUser.ObjectType);
        // код сценария по обработке activity
    }
    private Guid sysdbaUserGuid = new Guid("cad00016-306c-11d8-b4e9-00304f19f545");
}

```

11.9.2 Пример сценария типа «Сценарий для шагов ЖЦ».

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Intermech.Interfaces;

namespace Intermech.Kernel.Services.Scripts
{
    public class Script
    {
        /// <summary>
        /// Свойство ScriptContext для обращения к IPS API.
        /// Значение свойства заполняется автоматически непосредственно перед вызовом метода Execute.
        /// </summary>
        public ICSHarpScriptContext ScriptContext
        {
            get;
            set;
        }
        /// <summary>
        /// Метод вызываемый при изменении шага ЖЦ
        /// </summary>
        /// <param name="sender">Объект шаг ЖЦ которого изменяется</param>
        /// <param name="nextstep">Следующий шаг ЖЦ</param>
        /// <param name="session">Сессия</param>
        public void Execute(IDBObject sender, IDBLifecycleStep nextstep, IUserSession session)
        {
            throw new Exception("Error1");
        }
    }
}

```

11.9.3 Пример сценария «Сценарий для кнопок на формах ввода данных».

```

using System;
using System.ComponentModel.Design;
using System.Collections;
using System.Collections.Generic;
using System.Windows.Forms;
using Intermech.Interfaces;
using Intermech.Interfaces.Client;
public class Script
{
    /// <summary>
    /// Свойство ScriptContext для обращения к IPS API.
    /// Значение свойства заполняется автоматически непосредственно перед вызовом метода Execute.
    /// </summary>
    public ICSHarpScriptContext ScriptContext { get; private set; }
    // parameters содержит список значений атрибутов объекта, связи,
    // идентификаторы версий объекта и связи, а также пользовательскую сессию.
}

```

```
public AttributeValidationScriptParameters Execute(AttributeValidationScriptParameters parameters)
{
    // Получаем список значений атрибутов объекта из переданных параметров
    List<AttributeValues> attrValuesList = parameters.ObjectAttributeValues;
    // Ищем нужный атрибут по наименованию
    AttributeValues attrValues = attrValuesList.Find(x => x.AttributeName == "Наименование");
    // Изменяем значение атрибута
    attrValues.Values = new object[] { "Вал" };
    return parameters;
}
}
```

Это инструкция по использованию IPS SDK для IPS 9 или новее. Пожалуйста, перед началом работы прочтите инструкцию полностью.

12.1 Критические изменения

Начиная с IPS 8 используется новое SDK, несовместимое с SDK предыдущих версий IPS. Это связано с тем, что в серверной части IPS 8 появилась поддержка .NET 6 и работы на Linux. Для этого потребовалось адаптировать исходный код IPS под требования .NET 6 SDK и Visual Studio 2022, которые опираются на широкое использование NuGet-пакетов, обновленный формат csproj-проектов, а также на повсеместное редактирование csproj-проектов вручную.

Новое SDK устанавливается и обновляется вместе с IPS и не требует запуска отдельного инсталлятора. Теперь, просто установив или обновив IPS, вы можете быть уверены, что используете актуальную версию SDK.

Старое и новое SDK могут быть установлены на компьютере одновременно и не будут мешать друг другу.

12.2 Системные требования

Требования к компьютеру разработчика:

- Архитектура CPU: 64 бита.
- Число ядер CPU: 4 или больше.
- Оперативная память: 12 Gb или больше.
- Система хранения: SSD-диск с емкостью, достаточной для установки на него ОС, компиляторов, IDE и исходных кодов проектов.

Требования к ПО:

- Windows 10 версии 1909 или новее, Windows Server 2016 или новее в редакции Standard или Datacenter.
- IPS версии 9 или новее.
- Разработка на C#:
 - Visual Studio 2022 (только Windows) или JetBrains Rider 2023.3.3 (кросс-платформенный)
- Разработка на Typescript:
 - Node.js версии 20.18 LTS;
 - npm версии 10.8.2 (поставляется в составе Node.js);
 - Visual Studio 2022 (только Windows) или VSCode (кросс-платформенный).

12.3 Разработка на C#

12.3.1 Подготовка окружения разработчика на C#

На компьютере разработчика IPS должен быть установлен локально в варианте **сервер+клиент**. Использование сетевых инсталляций IPS не поддерживается.

Все шаблоны проектов и примеры использования IPS SDK рассчитаны на то, что IPS установлен в расположение по умолчанию - **C:\Program Files\IPS**, и порты сервера приложений также оставлены по умолчанию - **8008** (для Remoting) и **8080** (для WebAPI). Если вы использовали другое расположение или номера портов сервера приложений, то при создании проектов с помощью IPS SDK необходимо будет корректировать их настройки.

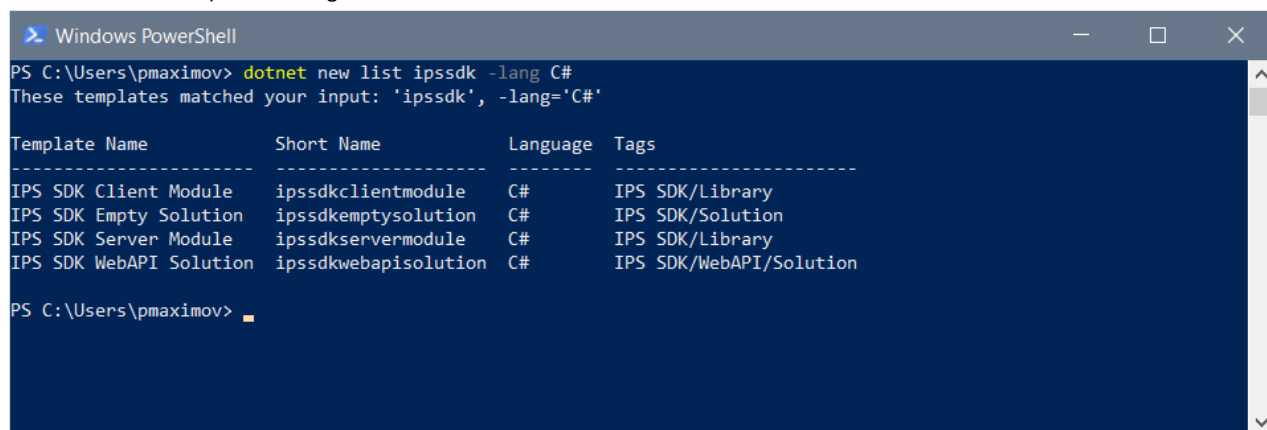
Основной IDE для разработки на C# является Visual Studio, так как она является наиболее удобной. VSCode (Visual Studio Code) на данный момент не поддерживается, так как она не обладает необходимыми возможностями по интеграции со сценариями MSBuild и по отладке Library-проектов, которыми являются все типы модулей расширения IPS.

12.3.2 Шаблоны проектов на C#

Для корректной работы IPS SDK требуется использовать только специальные шаблоны проектов и солюшенов на C#, разработанные в Intermech. Шаблоны проектов из Visual Studio или .NET SDK работать не будут.

Специальные шаблоны проектов на C# сами являются частью IPS SDK, они устанавливаются и обновляются вместе с IPS. Доступ к ним осуществляется через утилиту командной строки **dotnet**. Посмотреть список всех шаблонов и их краткие описания можно с помощью следующей команды в консоли ОС:

```
dotnet new list ipssdk -lang C#
```

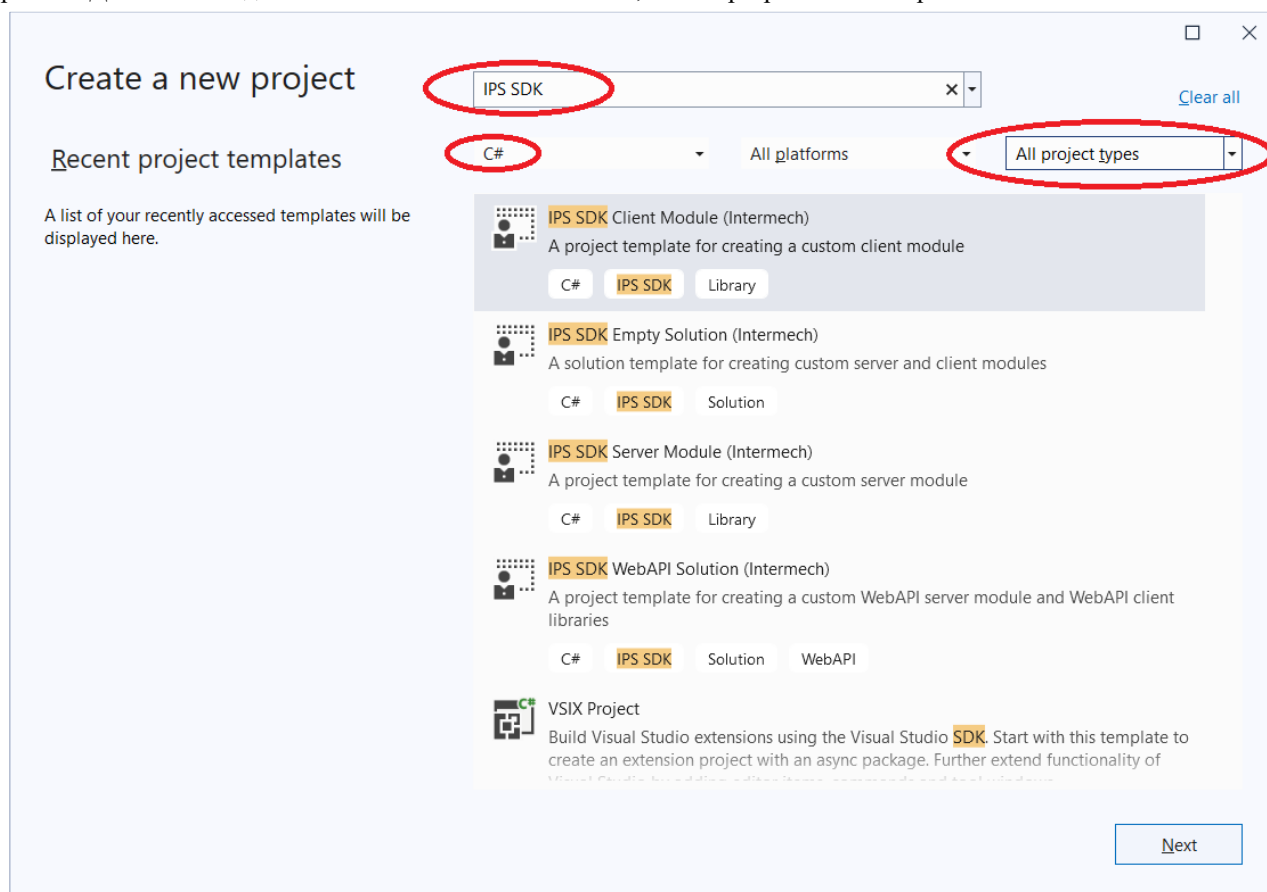


```
PS C:\Users\pmaximov> dotnet new list ipssdk -lang C#
These templates matched your input: 'ipssdk', -lang='C#'

Template Name                Short Name                Language  Tags
-----
IPS SDK Client Module        ipssdkclientmodule        C#        IPS SDK/Library
IPS SDK Empty Solution        ipssdkemptysolution        C#        IPS SDK/Solution
IPS SDK Server Module         ipssdkservermodule         C#        IPS SDK/Library
IPS SDK WebAPI Solution       ipssdkwebapisolution        C#        IPS SDK/WebAPI/Solution

PS C:\Users\pmaximov>
```

При работе в Visual Studio доступ к шаблонам осуществляется через стандартный диалог создания нового проекта. Для этого введите в поле поиска текст **IPS SDK**, а язык разработки выберите **C#**:



12.3.3 Создание нового солюшена

Внимание! Не используйте пробелы в именах создаваемых проектов, солюшенов или папок на диске. Из-за ошибки в .NET SDK создание проектов будет работать неправильно.

12.3.3.1 Visual Studio

При работе в Visual Studio:

2. Запустить Visual Studio.

3. На стартовом экране выбрать "Create a new project".
4. В поле "Search for templates" ввести "IPS SDK", либо в выпадающем списке "Project types" выбрать "IPS SDK".
5. Выбрать шаблон солюшена.
6. Заполнить имя солюшена (например, MyIPSMModule).
7. Убедиться, что установлен флажок "Place solution and project in the same directory".
8. Завершить создание, нажав кнопку "Create".

В результате в Visual Studio будет открыт созданный солюшен. Теперь можно переходить к конфигурированию IPS SDK для солюшена.

12.3.3.2 JetBrains Rider

При работе в JetBrains Rider следует учитывать, что он некорректно обрабатывает шаблоны солюшенов из-за ошибки **RIDER-71014** (<https://youtrack.jetbrains.com/issue/RIDER-71014>). Обойти эту ошибку можно следующим способом:

1. Запустить Rider.
2. На стартовом экране выбрать "New Solution".
3. В поле поиска ввести "IPS SDK".
4. Выбрать шаблон.
5. Заполнить имя солюшена в поле "Solution name" (например, MyIPSMModule).
6. Заполнить поле "Project name", указав то же имя, что и для солюшена (например, MyIPSMModule).
7. Убедиться, что установлен флажок "Put solution and project in the same directory".
8. Завершить создание, нажав кнопку "Create".

В процессе создания солюшена Rider покажет диалоговое окно с ошибкой Solution 'MyIPSMModule.sln' already exists. Это диалоговое окно следует просто закрыть. На самом деле солюшен MyIPSMModule.sln был успешно создан, и его можно открыть в Rider с диска.

Теперь можно переходить к конфигурированию IPS SDK для солюшена.

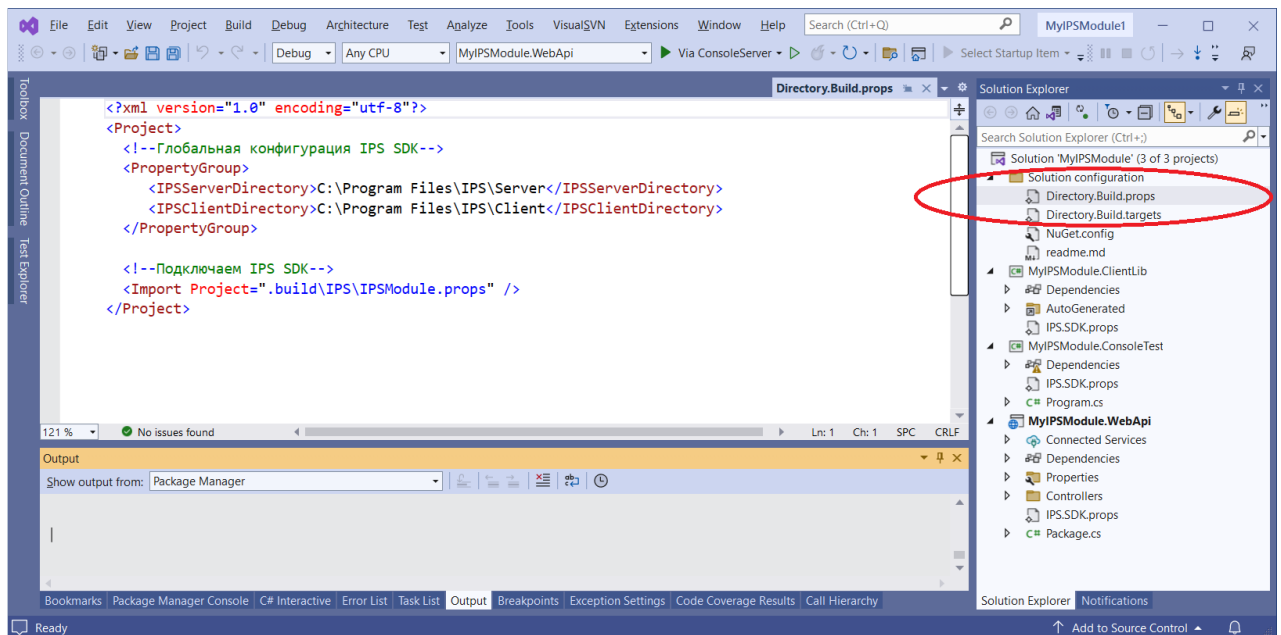
12.3.4 Добавление проектов в солюшен

Добавление проекта выполняется аналогично созданию нового солюшена: используя шаблоны проектов IPS SDK. Главное, чтобы и сам солюшен был создан по шаблону. Если же использовать пустой солюшен, который по умолчанию создает IDE, то потом невозможно будет сконфигурировать IPS SDK.

Внимание! Не используйте пробелы в именах создаваемых проектов, солюшенов или папок на диске. Из-за ошибки в .NET SDK создание проектов будет работать неправильно.

12.3.5 Конфигурирование IPS SDK для солюшена

После создания солюшена из шаблона необходимо сконфигурировать IPS SDK. Это делается с помощью файла Directory.Build.props, который расположен в корневой папке солюшена. В IDE этот файл можно найти в папке Solution configuration в дереве солюшена.



Файлы **Directory.Build.props** и **Directory.Build.targets** не являются частью IPS SDK. Они относятся к MSBuild и Visual Studio, являются сценариями MSBuild и используются для расширения стандартных возможностей MSBuild. Подробнее о них можно прочитать здесь:

<https://learn.microsoft.com/en-us/visualstudio/msbuild/customize-your-build?view=vs-2022>

Подробнее о формате файлов сценариев MSBuild можно прочитать здесь:

1. <https://learn.microsoft.com/en-us/visualstudio/msbuild/walkthrough-creating-an-msbuild-project-filefrom-scratch?view=vs-2022>
2. <https://learn.microsoft.com/en-us/visualstudio/msbuild/msbuild-project-file-schema-reference?view=vs-2022>

В файле **Directory.Build.props** следует указать значения для следующих параметров:

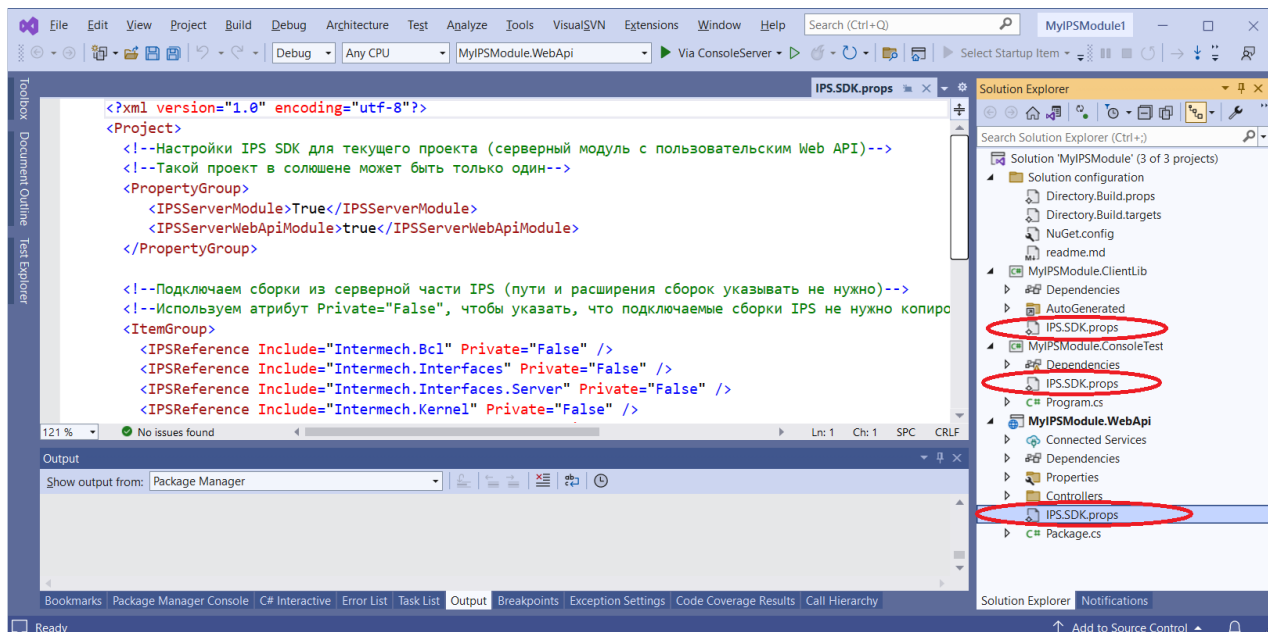
1. **IPSServerDirectory** - путь к папке с установленной серверной частью IPS.
2. **IPSClientDirectory** - путь к папке с установленной клиентской частью IPS.

Внимание! Если эти параметры не заданы или заданы неверно, то солюшен собираться не будет.

После изменения параметров в файле **Directory.Build.props** его следует просто сохранить. IDE автоматически подхватит изменения и применит их к солюшену.

12.3.6 Конфигурирование проектов в солюшене

Каждый проект модуля расширения IPS в солюшене содержит файл **IPS.SDK.props**. Он позволяет задать тип модуля IPS, а также указать ссылки на используемые сборки IPS. Этот файл также является сценарием MSBuild и имеет тот же формат, что и файлы **Directory.Build.props** и **Directory.Build.targets**.



После любых изменений в файле **IPS.SDK.props** его следует просто сохранить. IDE автоматически подхватит изменения и применит их к проекту.

12.3.6.1 Тип модуля расширения

Чтобы указать тип модуля расширения IPS создайте один из следующих параметров со значением True:

1. **IPSServerModule** - серверный модуль расширения.
2. **IPSClientModule** - клиентский модуль расширения.

Внимание! Не указывайте оба параметра одновременно.

Пример для серверного модуля расширения

```
<PropertyGroup>
<IPSServerModule>True</IPSServerModule>
</PropertyGroup>
```

Пример для клиентского модуля расширения

```
<PropertyGroup>
<IPSClientModule>True</IPSClientModule>
</PropertyGroup>
```

12.3.6.2 Ссылки на сборки IPS

Для создания ссылки на сборку IPS создайте элемент *IPSReference* с атрибутом *Include* в группе *ItemGroup*. В атрибуте *Include* укажите имя сборки без пути и без расширения.

Дополнительно у элемента *IPSReference* может быть задан необязательный атрибут *Private*, который управляет копированием сборки IPS в папку модуля расширения при компиляции:

- Если атрибут *Private* не задан или имеет значение True, то сборка IPS будет скопирована в папку модуля расширения.
- Если же атрибут *Private* задан и имеет значение False, то сборка IPS не будет скопирована в папку модуля расширения.

Например:

```
<ItemGroup>
<!--Эти сборки используются модулем расширения, но не копируются при компиляции
в папку модуля-->
<IPSReference Include="Intermech.Bcl" Private="False" />
<IPSReference Include="Intermech.Interfaces" Private="False" />
<!--Эти сборки используются модулем расширения и копируются при компиляции в
папку модуля-->
<IPSReference Include="ICSharpCode.SharpZipLib" />
```

</ItemGroup>

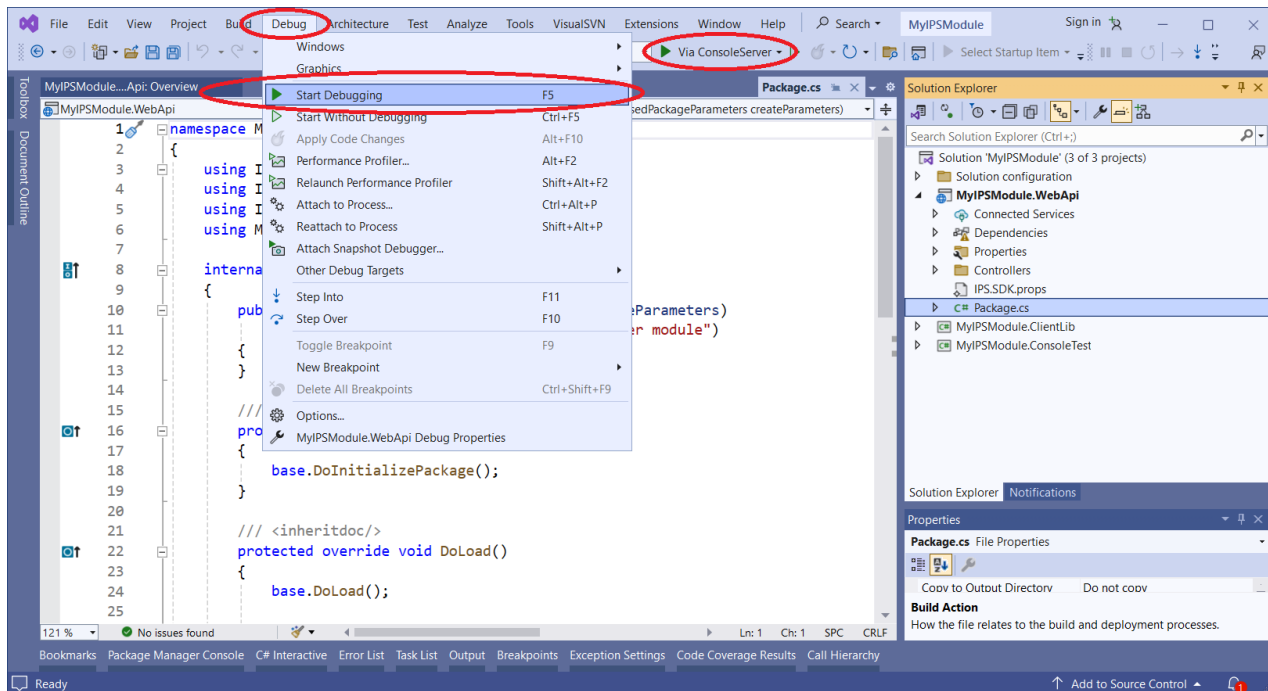
12.3.7 Отладка проектов в солюшене

Шаблоны проектов IPS SDK предоставляют простой и удобный способ отладки кода для всех типов модулей расширения IPS. Больше не требуется использовать post-build event для копирования .dll-файлов модулей расширения в папки клиента или сервера IPS. Также для отладки модулей расширения больше не требуется создавать объекты типа "Загружаемый модуль" в базе данных IPS.

12.3.7.1 Отладка одного модуля расширения

В Visual Studio для начала отладки любого модуля расширения IPS достаточно воспользоваться обычной командой Debug\Start Debugging из главного меню (горячая клавиша - F5), или нажать аналогичную кнопку на панели инструментов.

Кнопка на панели инструментов меняет свое название в зависимости от типа модуля расширения. Для серверных модулей она будет называться **Via ConsoleServer**, а для клиентских модулей - **Via IMClient**.

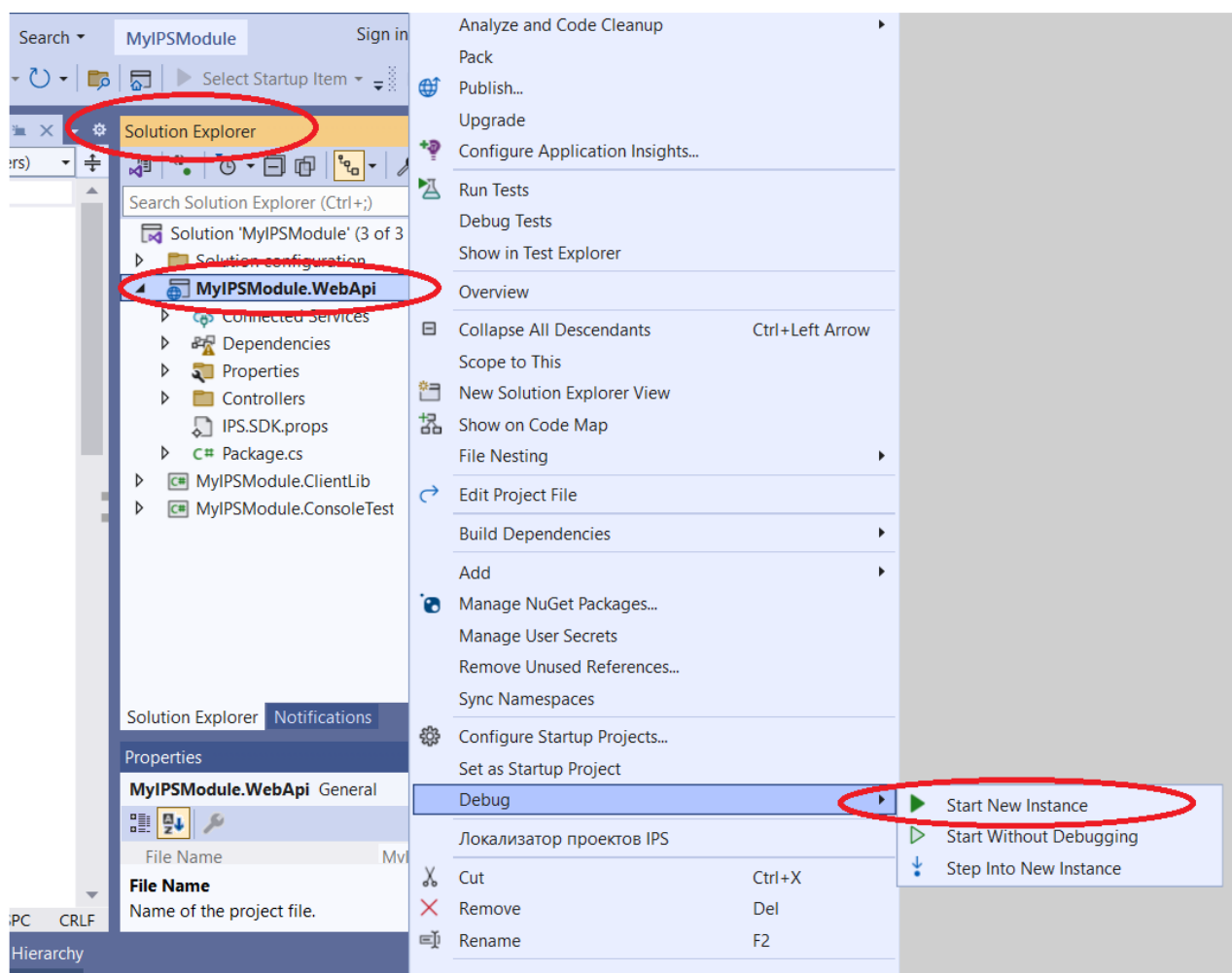


После запуска отладки будет сделано следующее:

1. Visual Studio запустит основное приложение IPS: ConsoleServer или IMClient.
2. Основное приложение IPS загрузит отлаживаемый модуль расширения непосредственно из папки **bin\Debug** или **bin\Release**, в которую его выкладывает Visual Studio в результате сборки.

12.3.7.2 Отладка нескольких модулей расширения одновременно

Если же ваш солюшен содержит серверные и клиентские модули расширения, которые необходимо отлаживать одновременно, воспользуйтесь командой контекстного меню **Debug\Start New Instance** из дерева солюшена (**Solution Explorer**). Сначала запустите отладку серверного модуля расширения, а затем клиентского.



12.3.8 Особенности пользовательских WebAPI-модулей

IPS SDK позволяет создавать серверные модули расширения, которые содержат пользовательское WebAPI. Для этого следует воспользоваться шаблоном солюшена "IPS SDK WebAPI Solution".

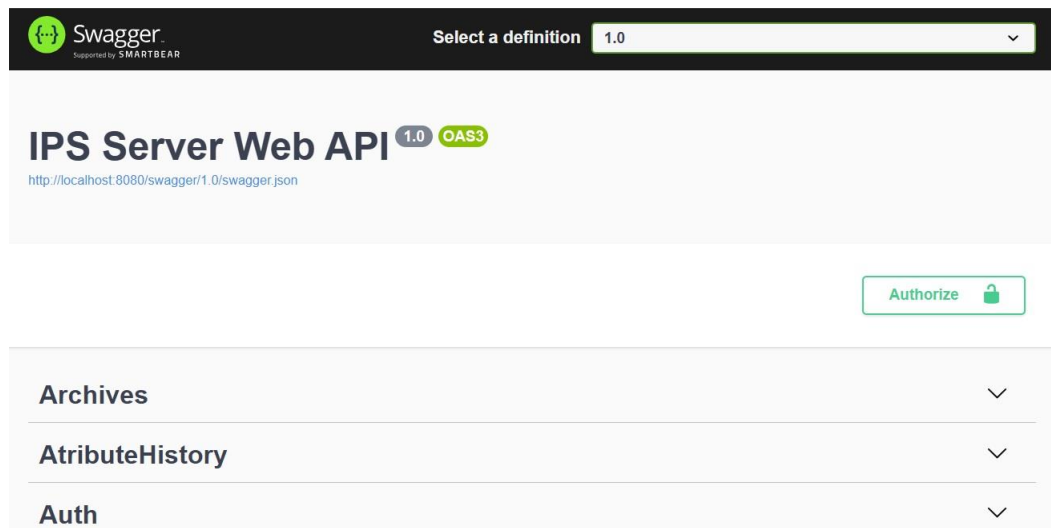
Например, если по этому шаблону создать солюшен MyIPSMModule, то он будет содержать 3 проекта:

1. MyIPSMModule.WebAPI - серверный модуль расширения с API-контроллерами ASP.NET.
2. MyIPSMModule.ClientLib - клиентская библиотека на C# для обращения к пользовательскому WebAPI.
3. IPSModule.ConsoleTest - пример консольного приложения, которое использует пользовательское WebAPI.

Внимание! В солюшене может быть только 1 серверный модуль расширения с пользовательскими API-контроллерами ASP.NET.

12.3.9 Просмотр списка доступных эндпоинтов и документации по WebAPI через Swagger UI

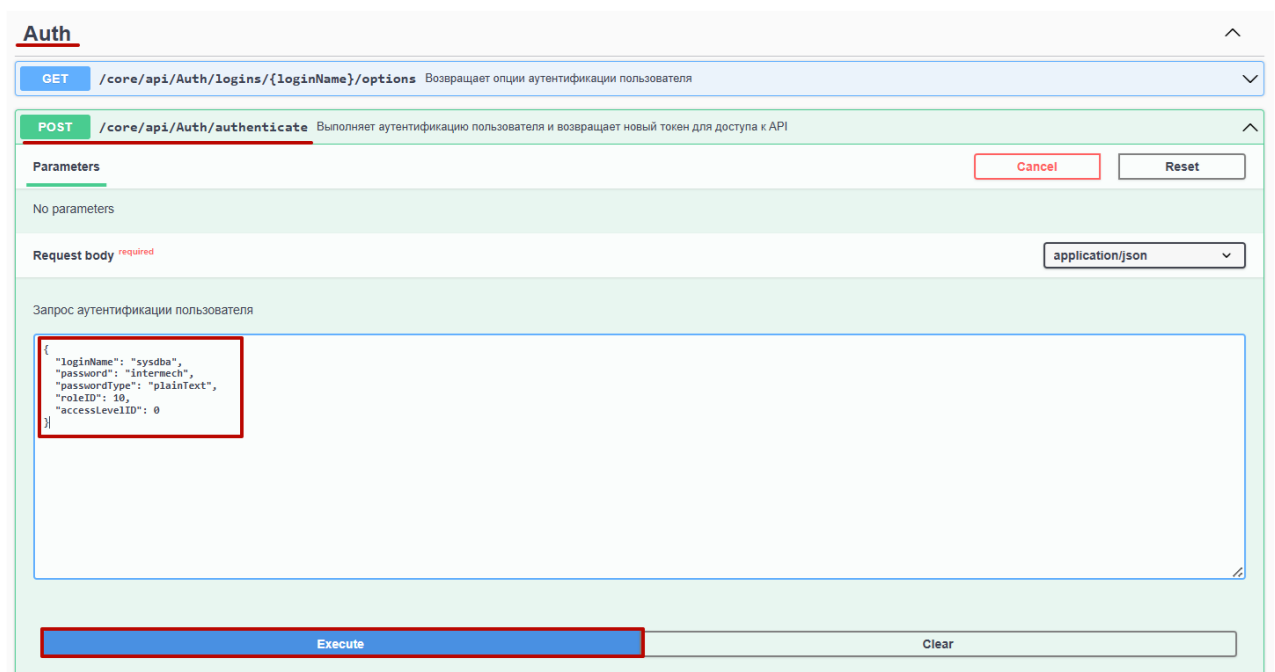
Страница Swagger UI для сервера, установленного по умолчанию и с портами по умолчанию доступна по ссылке: <http://localhost:8080/swagger/index.html>. В случае изменения настроек по умолчанию скорректируйте имя сервера и порт.



12.3.10 Взаимодействие с WebApi через Swagger UI

Большая часть доступных эндпоинтов требует авторизации. Для получения JWT-токена можно воспользоваться Swagger UI. Для этого необходимо вызвать эндпоинт **Auth/authenticate**.

1. Откройте в браузере страницу Swagger UI.
2. Разверните группу **Auth**.
3. В группе **Auth** разверните эндпоинт **/core/api/Auth/authenticate**.
4. Нажмите кнопку "Try it out".
5. В секции **Request body** заполните данные о пользователе.
6. Нажмите кнопку "Execute".



В результате выполнения запроса вернется пара токенов:

- **accessToken** - для доступа к API;
- **refreshToken** - для получения новой пары. Используется для продления времени действия токена доступа.

The screenshot shows the Swagger UI for the 'MyIPModule'. The 'Parameters' section has a 'value' field (string, path) with the value 'test'. The 'Execute' button is highlighted in blue. The 'Responses' section shows a 200 status code and a response body of 'test'. The 'Request URL' is 'http://localhost:8080/samples/api/MyIPModule/ping/test'. The 'Server response' is '200' with a 'Response body' of 'test'.

Запрос обработан успешно. Swagger UI корректно подставил **accessToken** в заголовок запроса.

12.3.11 Адаптация модулей расширения прежних версий IPS под новое IPS SDK

На сегодняшний день не существует каких-либо инструментов автоматической миграции исходного кода на новое IPS SDK для проектов модулей расширения, разработанных под прежние версии IPS. Всю работу приходится выполнять вручную.

Примерный план действий:

1. Создать новый пустой солюшен на базе шаблона "IPS SDK Empty Solution";
2. В новом солюшене создать заготовки всех проектов, как в существующем солюшене. Используйте шаблоны "IPS SDK Server Module" и "IPS SDK Client Module".
3. Для каждого нового проекта скопировать из существующего одноименного проекта все файлы исходного кода, ресурсов, ссылки на сборки IPS, настройки компиляции.
4. Скомпилировать и отладить проекты в новом солюшене.

12.4 Разработка на Typescript

12.4.1 Подготовка окружения разработчика на Typescript

На компьютере разработчика IPS должен быть установлен локально в варианте **только сервер**. Использование сетевых инсталляций IPS не поддерживается.

Все шаблоны проектов и примеры использования IPS SDK рассчитаны на то, что IPS установлен в расположение по умолчанию - **C:\Program Files\IPS**, и порты сервера приложений также оставлены по умолчанию - **8008** (для Remoting) и **8080** (для WebAPI). Если вы использовали другое расположение или номера портов сервера приложений, то при создании проектов с помощью IPS SDK необходимо будет корректировать их настройки.

Основной IDE для разработки на Typescript является VSCode (Visual Studio Code), так как она является наиболее удобной.

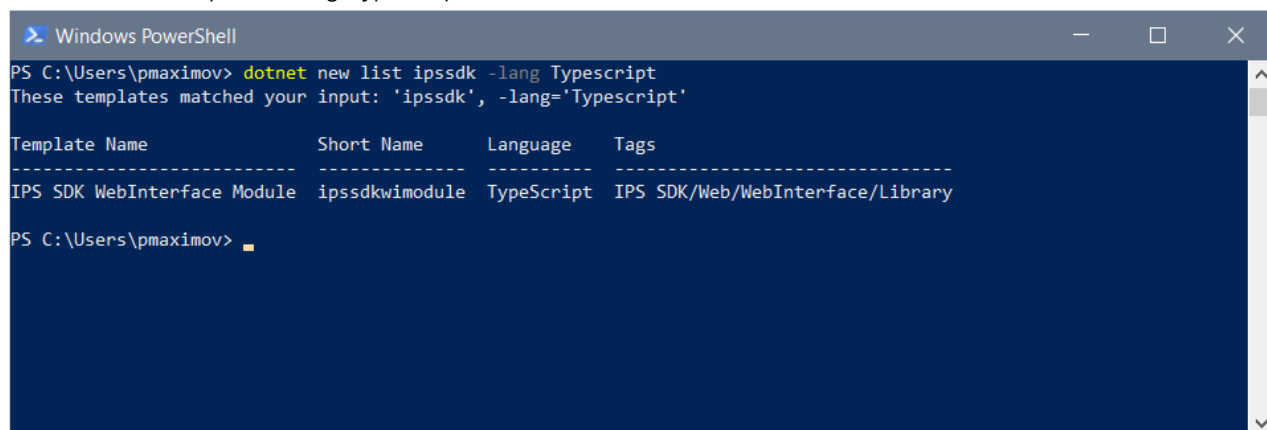
Также можно использовать и Visual Studio, если используется гибридный солюшен, в котором серверные модули расширения на C# сочетаются с клиентскими модулями на Typescript. Но следует помнить, что отладка Typescript-кода в Visual Studio будет работать только в браузерах на основе Chrome, а отладка в Firefox работать не будет.

12.4.2 Шаблоны проектов на Typescript

Для корректной работы IPS SDK требуется использовать только специальные шаблоны проектов на Typescript, разработанные в Intermech. Шаблоны проектов из VSCode, Visual Studio или npm работать не будут.

Специальные шаблоны проектов на Typescript сами являются частью IPS SDK, они устанавливаются и обновляются вместе с IPS. Как и в случае с проектами на C#, доступ к шаблонам на Typescript осуществляется через утилиту командной строки **dotnet**. Посмотреть список всех шаблонов и их краткие описания можно с помощью следующей команды в консоли ОС:

```
dotnet new list ipssdk -lang Typescript
```

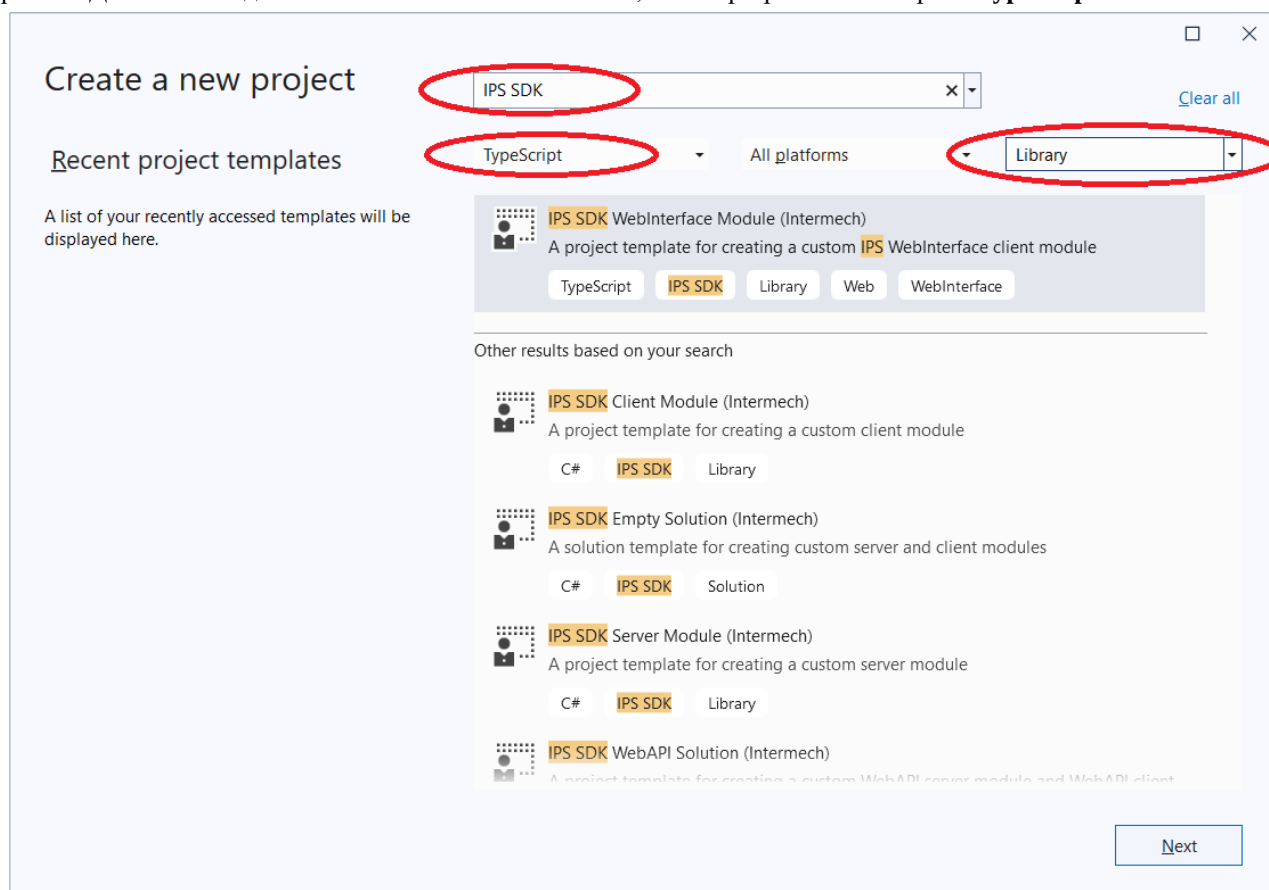


```
Windows PowerShell
PS C:\Users\pmaximov> dotnet new list ipssdk -lang Typescript
These templates matched your input: 'ipssdk', -lang='Typescript'

Template Name                Short Name      Language      Tags
-----
IPS SDK WebInterface Module  ipssdkwimodule TypeScript    IPS SDK/Web/WebInterface/Library

PS C:\Users\pmaximov>
```

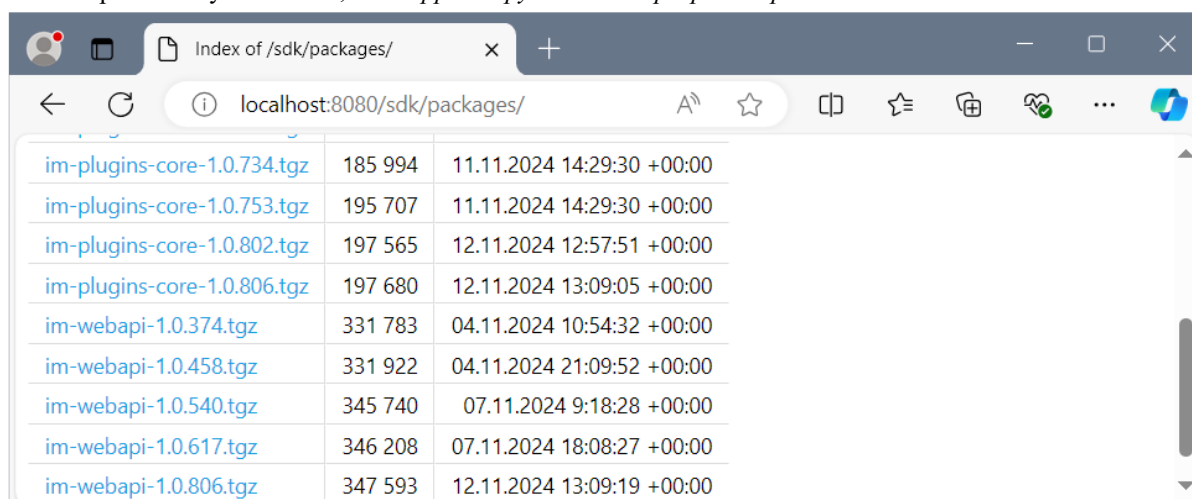
При работе в Visual Studio доступ к шаблонам осуществляется через стандартный диалог создания нового проекта. Для этого введите в поле поиска текст **IPS SDK**, а язык разработки выберите **Typescript**:



12.4.3 npm-пакеты для IPS SDK

Как это принято в мире Web-разработки, клиентское IPS API для Typescript поставляется в виде npm-пакетов. Они отличаются от традиционных пакетов тем, что репозиторием для npm-пакетов с IPS API является не <https://npmjs.com>, а сам сервер приложений IPS.

Вы можете увидеть все версии npm-пакетов, поддерживаемые установленной версией IPS, открыв в любом браузере ссылку вида <http://localhost:8080/sdk/packages>. Если IPS установлен не в расположение по умолчанию или не с портами по умолчанию, то *скорректируйте имя сервера и порт*:



im-plugins-core-1.0.734.tgz	185 994	11.11.2024 14:29:30 +00:00
im-plugins-core-1.0.753.tgz	195 707	11.11.2024 14:29:30 +00:00
im-plugins-core-1.0.802.tgz	197 565	12.11.2024 12:57:51 +00:00
im-plugins-core-1.0.806.tgz	197 680	12.11.2024 13:09:05 +00:00
im-webapi-1.0.374.tgz	331 783	04.11.2024 10:54:32 +00:00
im-webapi-1.0.458.tgz	331 922	04.11.2024 21:09:52 +00:00
im-webapi-1.0.540.tgz	345 740	07.11.2024 9:18:28 +00:00
im-webapi-1.0.617.tgz	346 208	07.11.2024 18:08:27 +00:00
im-webapi-1.0.806.tgz	347 593	12.11.2024 13:09:19 +00:00

Итак, разработчику на Typescript под IPS требуется:

1. Доступ к <https://npmjs.com> - для получения пакетов typescript, webpack, react и т.д..
2. Доступ к работающему серверу приложений IPS - для получения пакетов IPS API.

12.4.4 Создание клиентского модуля расширения для IPS WebInterface

Клиентский модуль расширения для IPS WebInterface - это аналог клиентского модуля на C#, загружаемого в IMClient. С его помощью вы сможете:

- добавлять новые команды для объектов IPS или узлов навигатора на панели инструментов и в контекстные меню;
- создавать новые виды навигатора;
- изменять или расширять поведение мастера создания объектов;
- обрабатывать события в UI или в клиентской бизнес-логике.

В клиентских модулях для IPS WebInterface для доступа к серверу приложений IPS вместо Remoting используется новое WebAPI.

Рассмотрим создание нового проекта в среде VSCode и Visual Studio.

Внимание! Не используйте пробелы в именах создаваемых проектов или папок на диске. Из-за ошибки в .NET SDK создание проектов будет работать неправильно.

12.4.4.1 VSCode

При работе в VSCode:

1. Запустить сервер приложений IPS, если это еще не было сделано.
2. Создать папку проекта на локальном диске. Имя папки будет являться и именем проекта, поэтому не используйте пробелы в имени. Например, пусть это будет папка **my-wi-module** по пути `c:\src\my-wi-module`.
3. Запустить VSCode, если это еще не было сделано. Все дальнейшие действия выполняются в VSCode.
4. Открыть вкладку **Terminal** с помощью команды **View/Terminal**.
5. На вкладке **Terminal** выполнить следующие команды для создания проекта:

```
cd c:\src\my-wi-module
dotnet new ipssdkwimodule
```

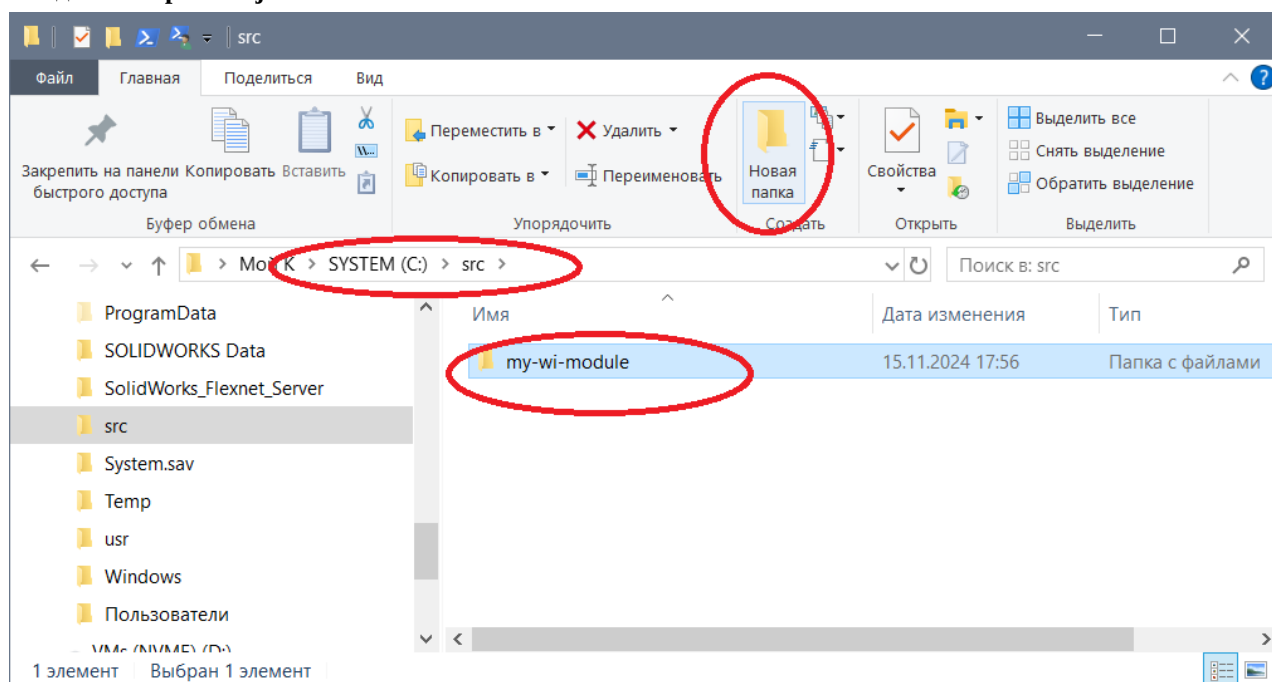
6. Открыть файл **my-wi-module.code-workspace** по пути `c:\src\my-wi-module` с помощью команды **File/Open Workspace from File**.
7. Если сервер приложений IPS установлен и запущен не в расположении по умолчанию (т.е. `http://localhost:8080`), то необходимо сделать следующее:
 - открыть файл **package.json** и исправить ссылки на npm-пакеты IPS SDK в tarball-формате. Такие ссылки начинаются на `http://localhost:8080/sdk/packages`;
 - открыть файл **.vscode/launch.json** и исправить ссылки на сервер приложений IPS в конфигурациях отладчика Typescript.
8. Перейти на вкладку **Terminal** и выполнить команду установки необходимых пакетов npm:

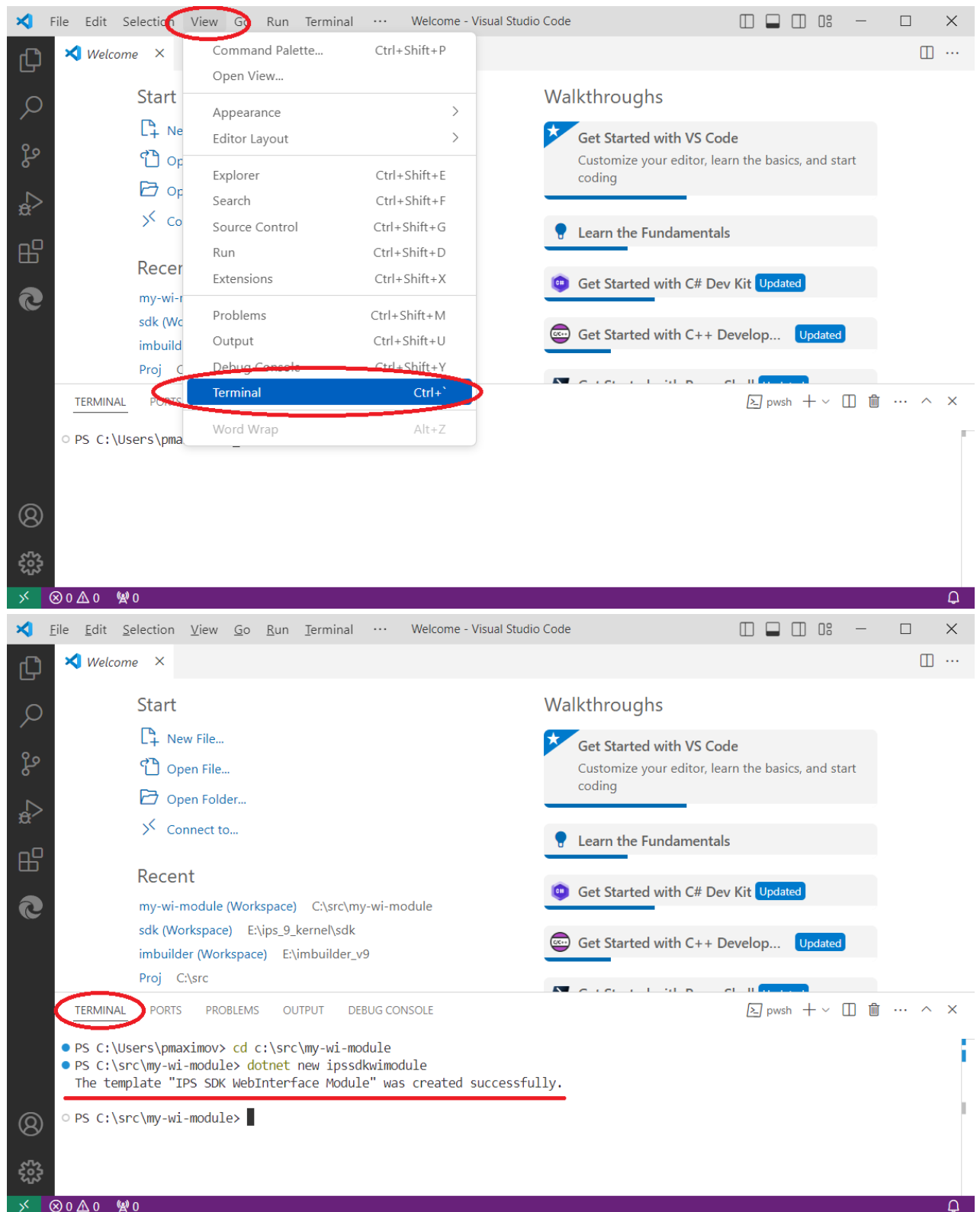
```
npm install
```

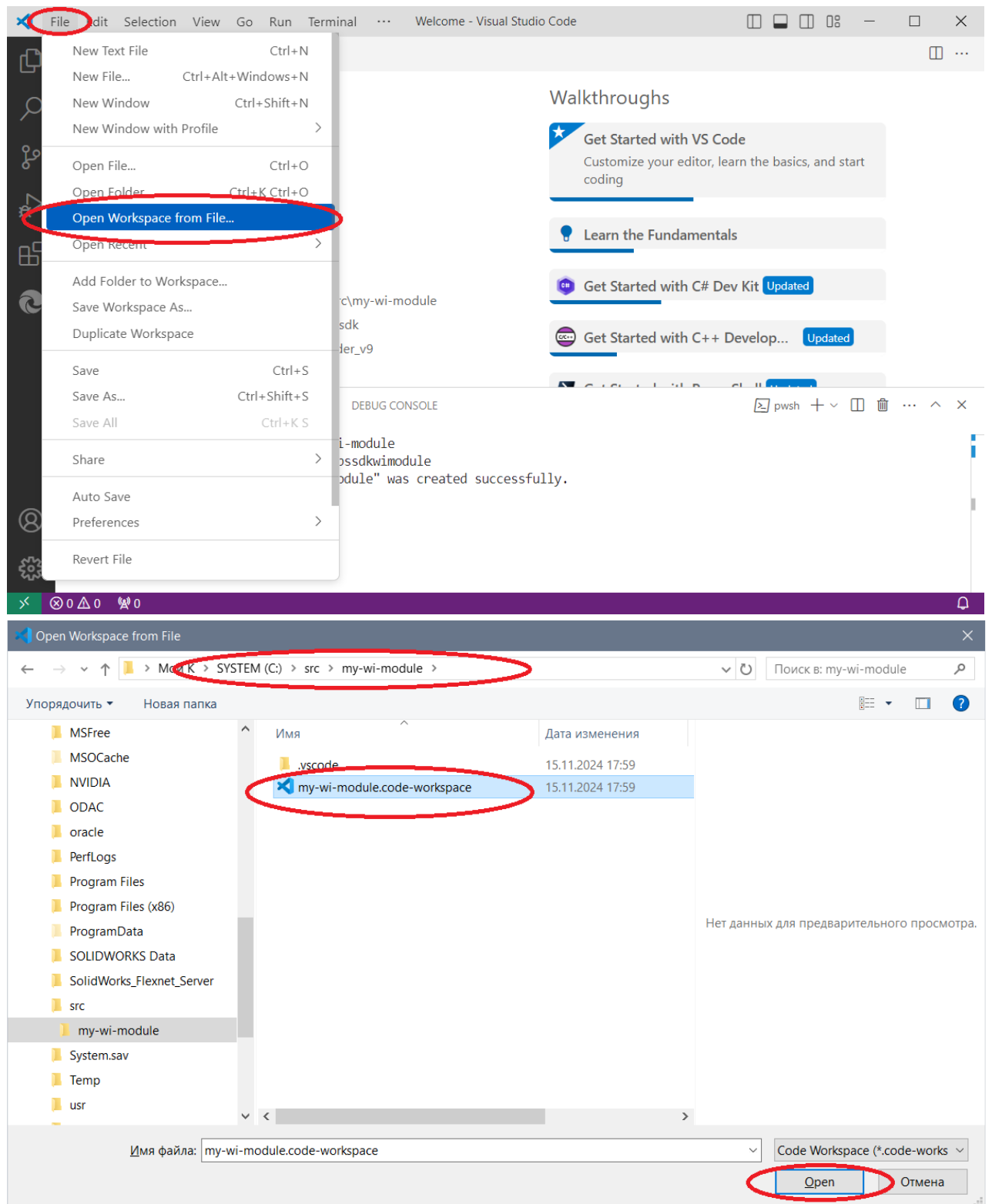
9. Выполнить команду **build**, чтобы убедиться, что проект успешно собирается. Сделать это можно либо с помощью таба **Explorer/NPM Scripts**, либо на вкладке **Terminal** с помощью команды:

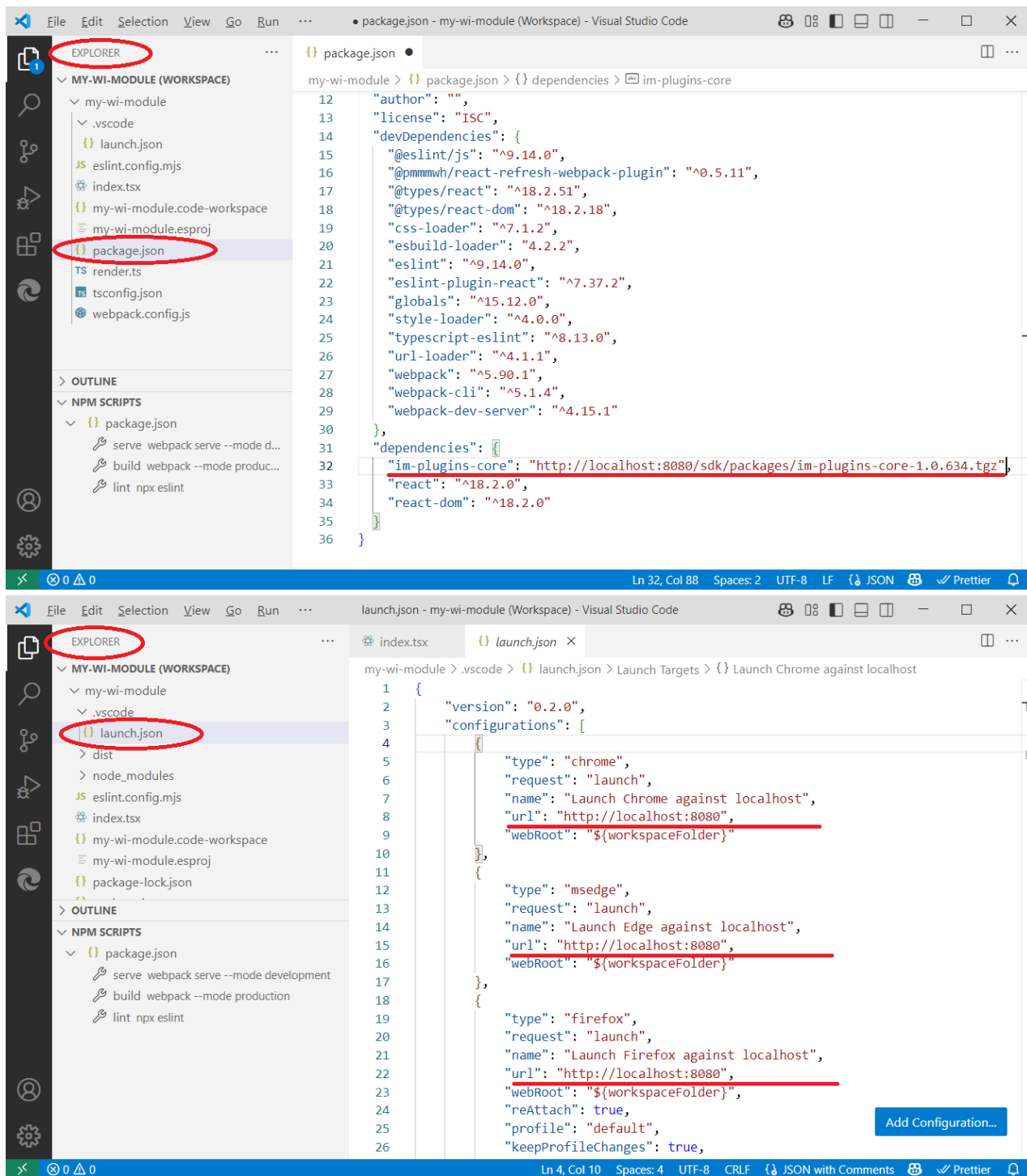
```
npm run build
```

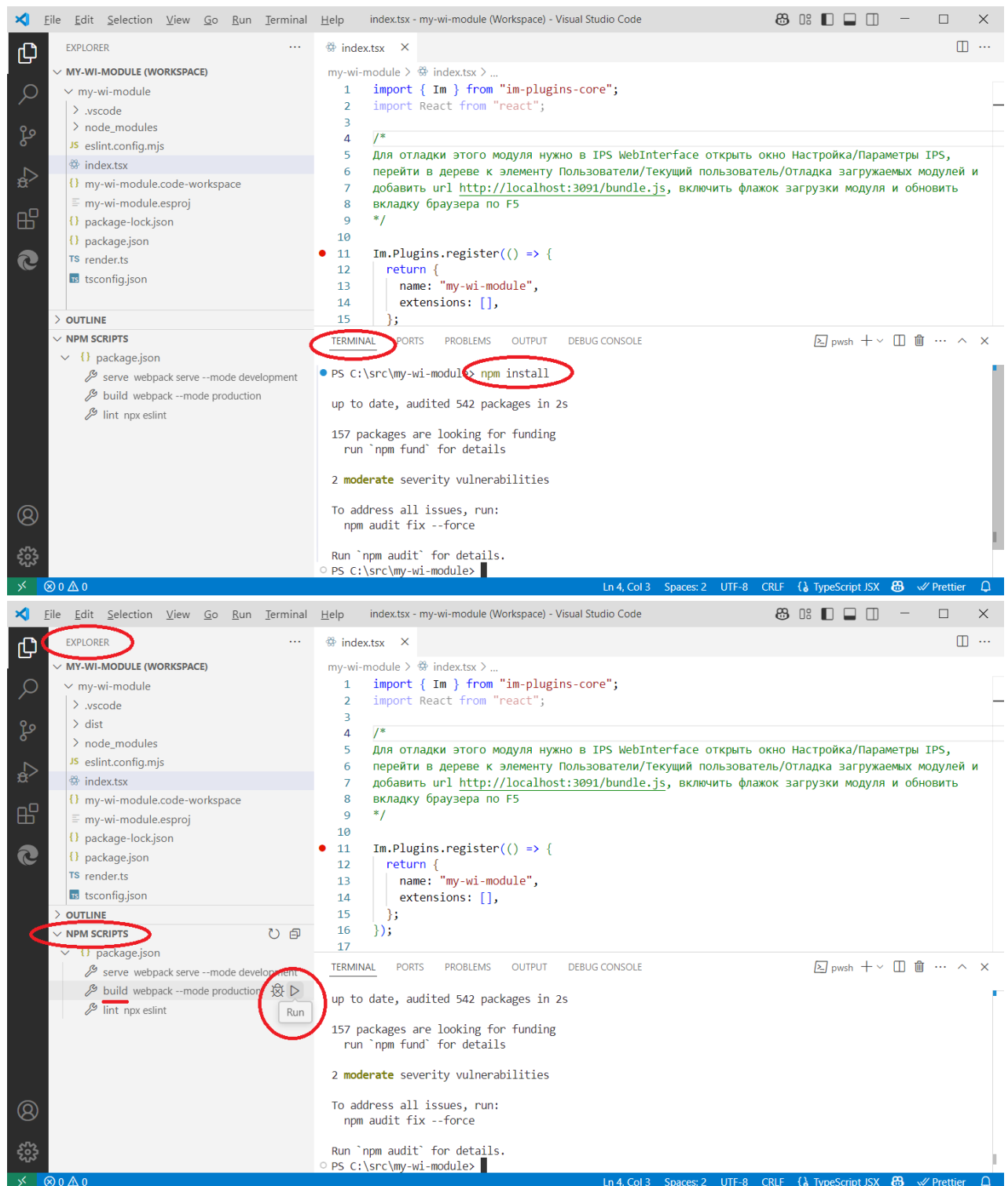
Обратите внимание, что хотя проект **my-wi-module** и был создан в среде VSCode, с ним можно работать и в Visual Studio. Для этого следует открыть файл **my-wi-module.esproj** по пути `c:\src\my-wimodule` с помощью команды **File/Open/Project/Solution**.

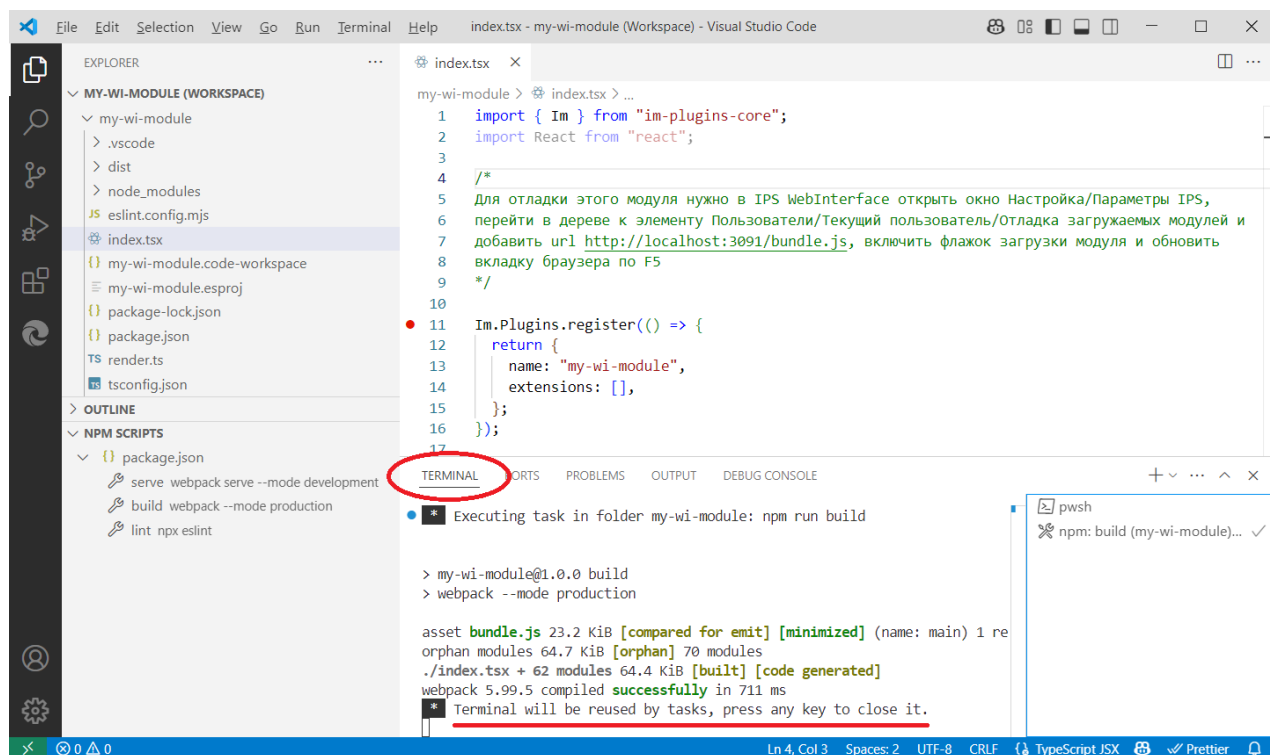










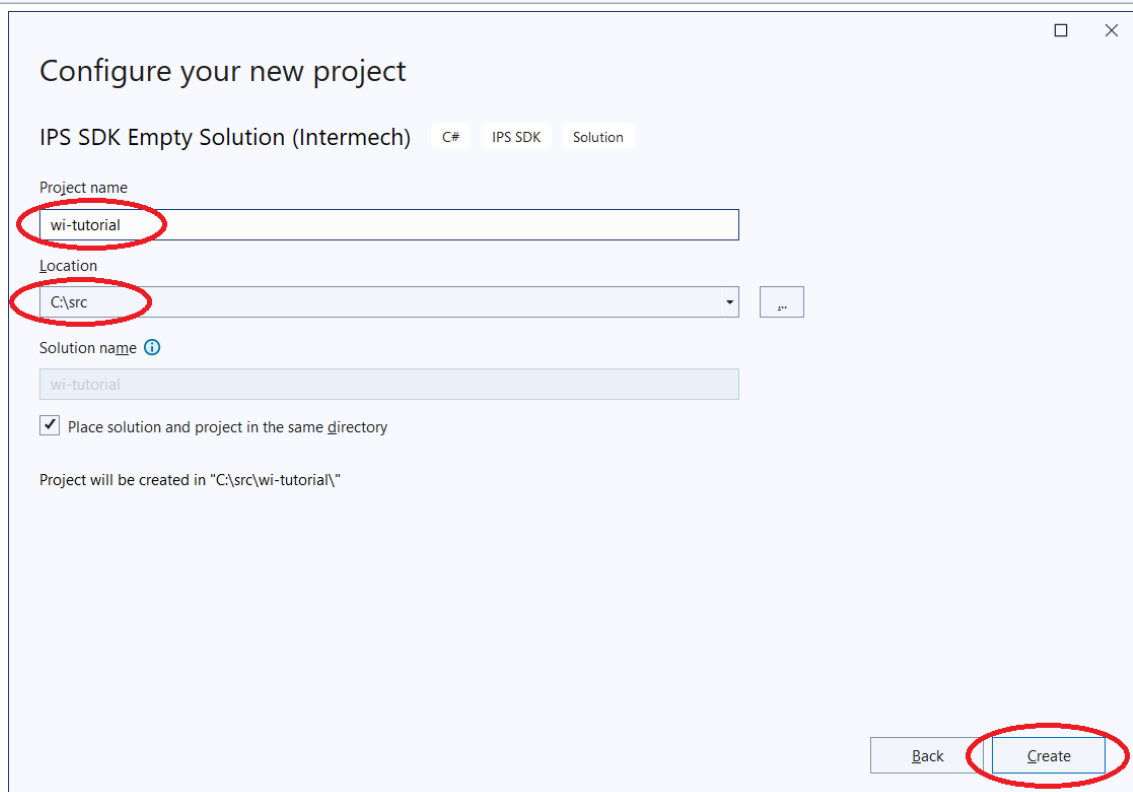
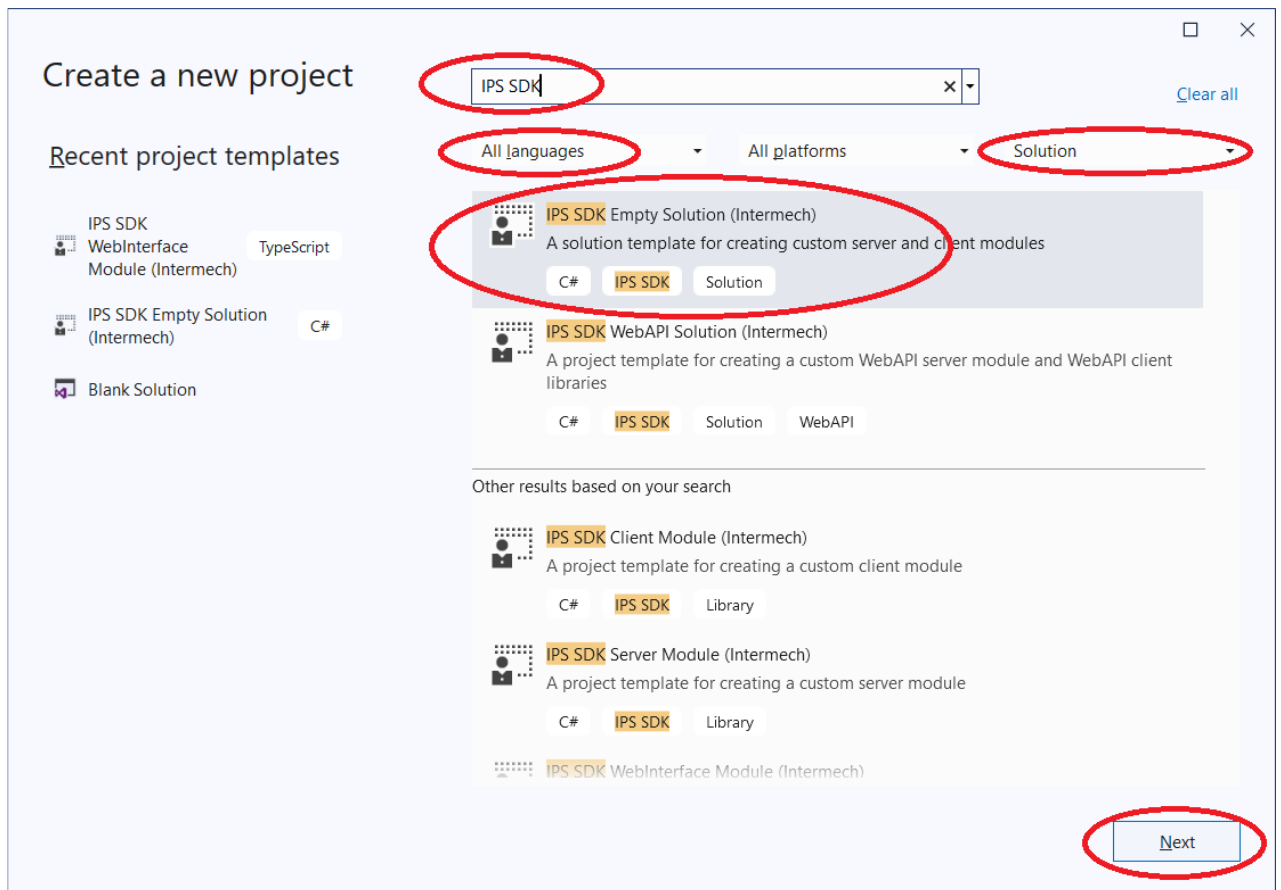


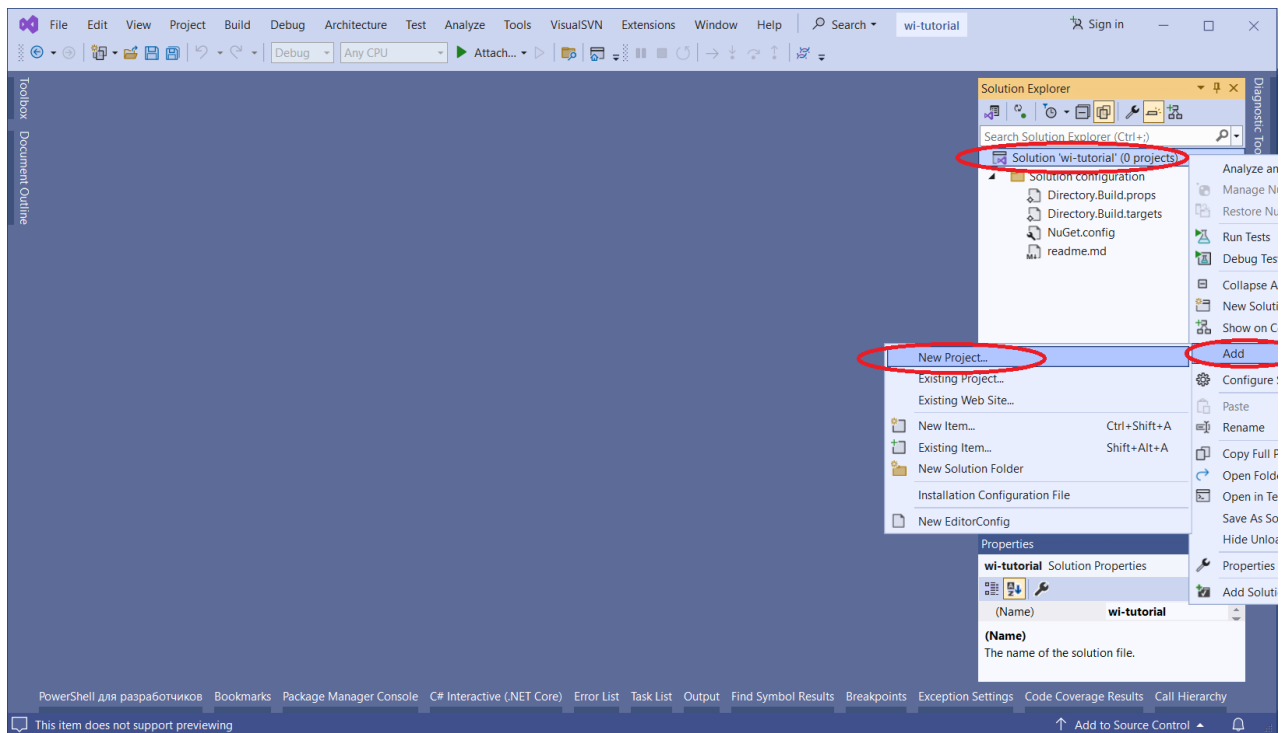
12.4.4.2 Visual Studio

При работе в Visual Studio:

1. Запустить сервер приложений IPS, если это еще не было сделано.
2. Запустить Visual Studio, если это еще не было сделано.
3. Открыть существующий солюшен или создать новый на базе шаблона **IPS SDK Empty Solution**.
4. Открыть мастер добавления нового проекта с помощью команды **Add/New Project** в дереве солюшена.
5. Выбрать шаблон **IPS SDK WebInterface Module** и указать имя проекта. Не используйте пробелы в имени проекта. Например, пусть это будет **my-wi-module**.
6. Если сервер приложений IPS установлен не в расположении по умолчанию (т.е. `http://localhost:8080`), то необходимо сделать следующее:
 - открыть файл **package.json** и исправить ссылки на npm-пакеты IPS SDK в tarball-формате. Такие ссылки начинаются на `http://localhost:8080/sdk/packages`;
 - открыть файл **.vscode/launch.json** и исправить ссылки на сервер приложений IPS в конфигурациях отладчика Typescript.
7. Выполнить команду **Project/Build**, чтобы убедиться, что проект успешно собирается.

Обратите внимание, что хотя проект **my-wi-module** и был создан в среде Visual Studio, с ним можно работать и в VSCode. Для этого следует открыть файл `my-wi-module.code-workspace` по пути `c:\src\mywi-module` с помощью команды **File/Open Workspace from File**.





Configure your new project

IPS SDK WebInterface Module (Intermech)TypeScriptIPS SDKLibraryWebWebInterface

Project name

my-wi-module

Location

C:\src\wi-tutorial

...

Project will be created in "C:\src\wi-tutorial\my-wi-module"

Back

Create

FileEditViewProjectBuildDebugArchitectureTestAnalyzeToolsVisualSVNExtensionsWindowHelpwi-tutorialSign in

DebugAny CPULaunch Chrome against localhost

launch.jsonpackage.json

Schema: https://json.schemastore.org/package.json

```
1 {
2   "name": "my-wi-module",
3   "version": "1.0.0",
4   "description": "",
5   "private": true,
6   "scripts": {
7     "serve": "webpack serve --mode development",
8     "build": "webpack --mode production"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "devDependencies": {
14    "@pmmwh/react-refresh-webpack-plugin": "^0.5.11",
15    "@types/react": "^18.2.51",
16    "@types/react-dom": "^18.2.18",
17    "esbuild-loader": "^4.0.3",
18    "webpack": "^5.90.1",
19    "webpack-cli": "^5.1.4",
20    "webpack-dev-server": "^4.15.1"
21  },
22  "dependencies": {
23    "im-plugins-core": "http://localhost:8080/sdk/packages/im-plugins-core-1.0.634.tgz",
24    "react": "^18.2.0",
25    "react-dom": "^18.2.0"
26  }
27 }
```

Solution Explorer

Search Solution Explorer (Ctrl+)

Solution 'wi-tutorial' (1 of 1 project)

Solution configuration

Directory.Build.props

Directory.Build.targets

NuGet.config

readme.md

my-wi-module

Dependencies

.vscode

launch.json

index.tsx

my-wi-module.code-workspace

package.json

render.ts

tsconfig.json

webpack.config.js

Solution ExplorerNotifications

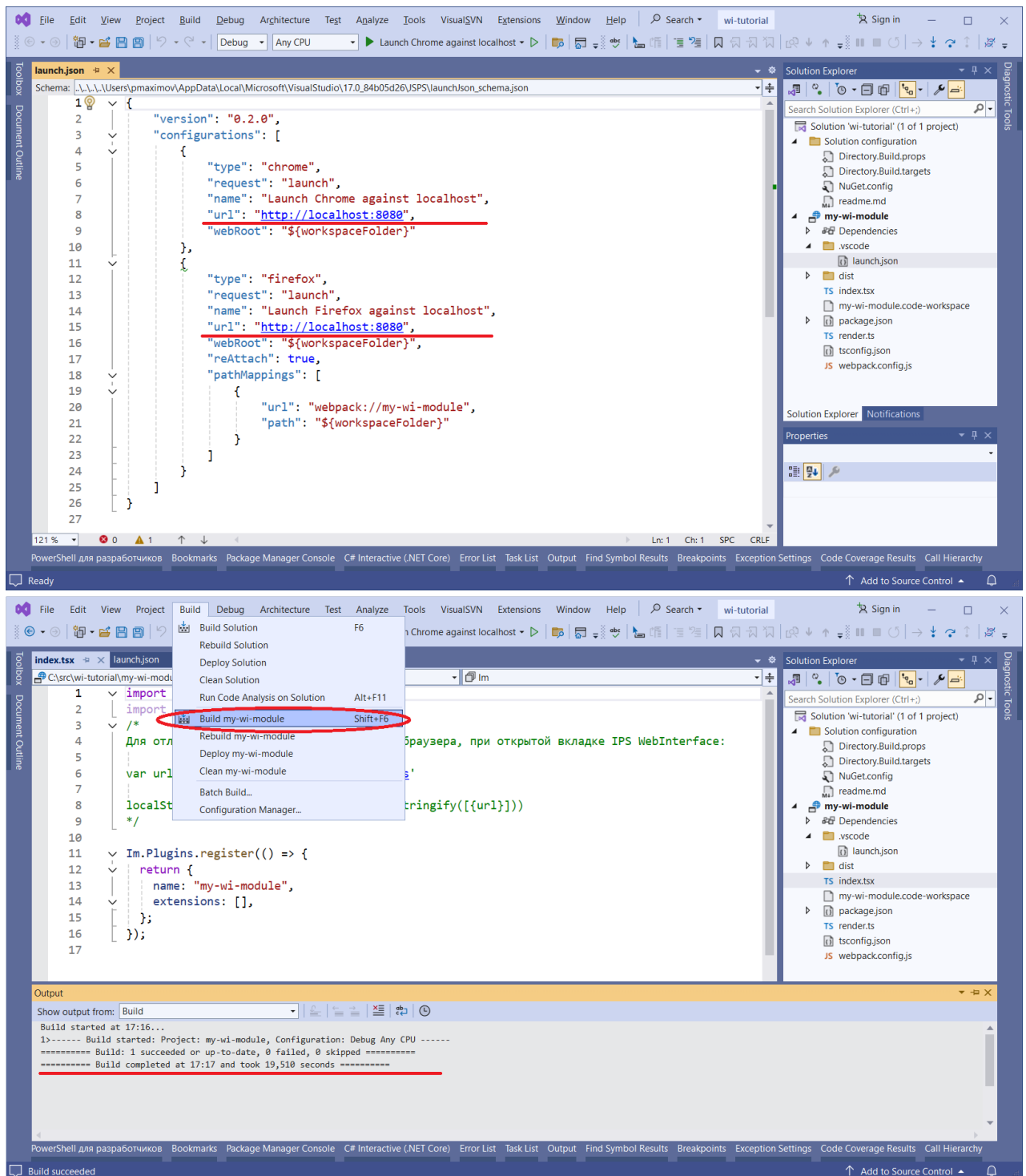
Properties

121 %No issues found

Ln: 1Ch: 1SPCLF

PowerShell для разработчиковBookmarksPackage Manager ConsoleC# Interactive (.NET Core)Error ListTask ListOutputFind Symbol ResultsBreakpointsException SettingsCode Coverage ResultsCall Hierarchy

ReadyAdd to Source Control



12.4.5 Отладка

В режиме отладки плагин работает как отдельное приложение, с которым IPS WebInterface взаимодействует по сети, используя адрес `http://localhost:3091/bundle.js`. Используемый для отладки порт **3091** не является жестко зафиксированным, и, при необходимости, порт можно изменить в файле `webpack.config.js`.

12.4.5.1 Запуск отладочного сервера

Первым шагом для отладки плагина всегда будет запуск отладочного сервера webpack с помощью команды `serve`.

При работе в **VSCode** это можно сделать либо с помощью таба **Explorer/NPM Scripts**, либо на вкладке **Terminal** с помощью команды:

```
npm run serve
```

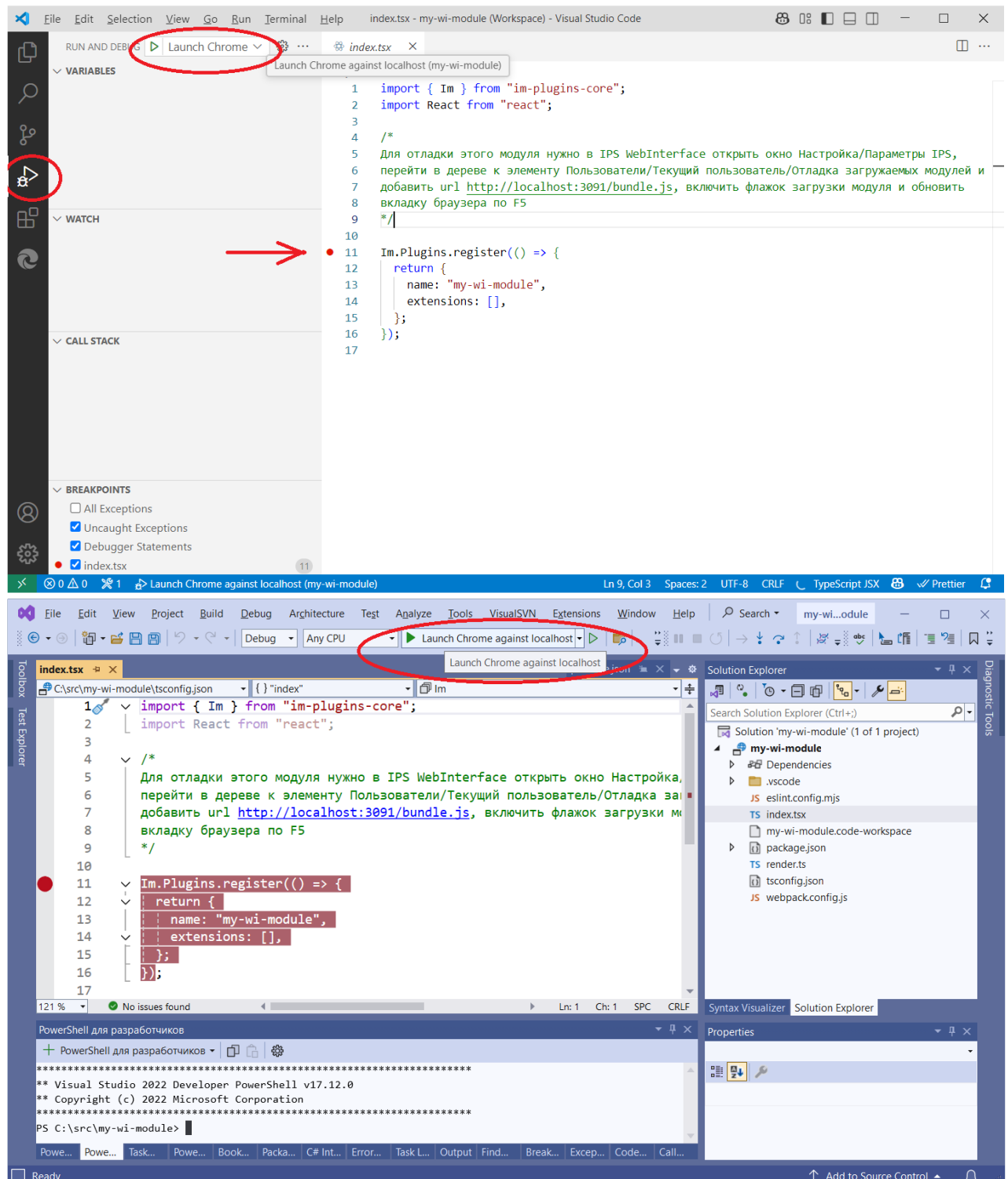
При работе в **Visual Studio** вкладку **Terminal** для плагина можно открыть с помощью контекстного меню проекта плагина в дереве солюшена, выбрав пункт **Open in Terminal**, а затем выполнить приведенную выше команду.

12.4.5.2 Запуск браузера в режиме отладки

Теперь можно запустить браузер в режиме отладки кода плагина. При работе в **VSCode** воспользуйтесь табом **Run**, в поле **Run and Debug** выберите профиль отладки для вашего браузера. При работе в **Visual Studio** выберите профиль отладки для вашего браузера на панели инструментов **Standard**.

В результате запустится выбранный браузер, в котором будет открыт IPS WebInterface.

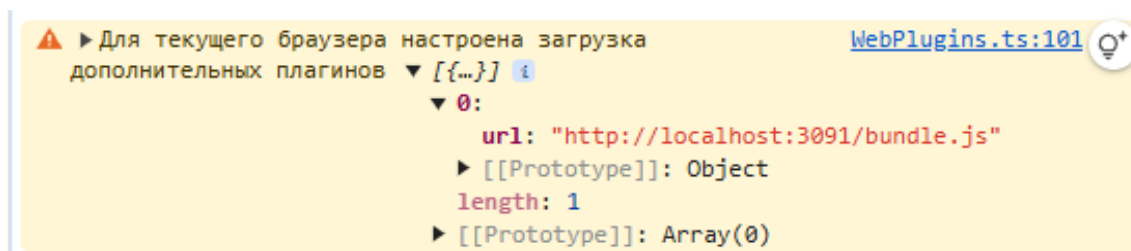
При отладке точки останова (breakpoints) можно расставлять непосредственно в IDE. Но также можно использовать и Dev Tools самого браузера.



12.4.5.3 Однократная настройка IPS WebInterface для отладки плагина

Также нужно однократно выполнить настройку IPS WebInterface:

1. В главном меню выбрать **Настройка** → **Параметры IPS**.
2. В папке **Пользователи** выбрать **Текущий пользователь** → **Отладка загружаемых модулей**.
3. С помощью кнопки + добавить пользовательский плагин, в поле **URL загружаемого модуля** ввести адрес `http://localhost:3091/bundle.js` и поставить флажок в колонке **Статус**.
4. Нажать кнопку «Применить».
5. Перезагрузить страницу. В консоли браузера появится следующее сообщение:



Приложение 1 Типы данных атрибутов IPS Development Kit

Перечислитель **FieldTypes** может принимать одно из значений:

- **ftString** – строка длиной до `Intermech.Consts.MaxStringSize`;
- **ftInteger** – целое число типа `Int64`;
- **ftDouble** – вещественное число;
- **ftDateTime** – дата и время (точность зависит от используемой СУБД);
- **ftShortBlob** – короткие двоичные данные длиной до `Intermech.Consts.MaxShortBlobSize`;
- **ftFile** – файл;
- **ftExternalLink** – внешняя ссылка (в данной версии IPS Development Kit не используется);
- **ftObjectLink** – ссылка на версию объекта;
- **ftPassword** – пароль;
- **ftMemo** – текст длиной до `Intermech.Consts.MaxShortBlobSize`;
- **ftBlob** – двоичные данные произвольной длины;
- **ftBoolean** – логический тип данных;
- **ftMeasured** – вещественное число, выраженное в единицах измерения;
- **ftAutoInc** – автоинкрементное целое число типа `Int64`;
- **ftGuid** – глобальный идентификатор.

Для выяснения возможности преобразования типа данных в другой тип можно воспользоваться методом `IDBAttributeType.IsCompatibleType`. Для проверки возможности присвоения значения другого атрибута служит метод `IDBAttributeType.ValidateAssign`.

Приложение 2 Стандартные команды контекстных меню

Команда контекстного меню – это один из элементов шаблона. У каждой команды есть несколько ключевых свойств:

- уникальное в пределах всей системы название команды;
- текстовое описание команды, которое видно как пункт контекстного меню;
- целочисленный индекс изображения для пункта меню (из глобального списка именованных изображений), значение **-1** означает отсутствие изображения;
- целочисленный номер группы, которой принадлежит команда;
- целочисленный номер команды внутри группы. По этому номеру команды сортируются перед своим отображением в составе контекстного меню;
- сочетание «горячих» клавиш, по которым можно вызвать данную команду в приложении.

Рассмотрим некоторые стандартные команды из шаблона контекстных меню:

Название команды	Текстовое описание команды	Название изображения	Номер группы	Номер в группе	«Горячие» клавиши
Открытие новых окон и другие операции над пользовательским интерфейсом					
Refresh	Обновить	imgRefresh	5	30	Keys.R Keys.Control
RestoreSelectionValues	Восстановить значения в условиях		5	31	
OpenInNewWindow	Открыть в новом окне	imgNavigator	5	40	Keys.Return Keys.Control
ParametersCard	Свойства(Карточка)	imgCard	5	50	Keys.F4
Find	Найти	imgSearch	5	60	
SeekInTree	Найти в дереве	imgSearchTree	5	65	Keys.Control Keys.F
ExpandNode	Развернуть		10	10	
ExpandNodeRecursive	Развернуть все		10	11	Keys.Alt Keys.Down
CollapseNode	Свернуть		10	12	
Создание объектов					
AddFolder	Создать папку	imgNewFolder	15	2	
Create	Создать		15	10	
CreateNew	Создать\Новый	imgNewItem	10	10	Keys.N Keys.Control
CreateInclude	Создать\В составе		10	15	
CreateProto	По прототипу		10	20	
CreateVersion	Создать\Версию		10	30	
CreateLinkedContext	Создать\Связанный контекст		10	100	

Название команды	Текстовое описание команды	Название изображения	Номер группы	Номер в группе	«Горячие» клавиши
SetVersionsRule	Установить правило подбора	imgVersionRuleEditor	10	31	
EditingContextActivate	Активизировать контекст	imgEditingContextsMode	10	32	
EditingContextAdd	Добавить в контекст	imgEditingContextsAdd	10	33	
EditingContextAddComposition	Добавить с составом	imgEditingContextsAddComposition	10	34	
AddIncludeWithLinkSelection	Добавить в состав с выбором связи	imgSnapshot	10	30	
SaveToDisk	Сохранить на диск		20	30	
CreateSnapshot	Создать \Итерацию		20	35	
CreateIncludeWithLinkSelection	Создать в составе с выбором связи	10	30		
RestoreSnapshot	Восстановить итерацию	1	1		
DeleteSnapshot	Удалить итерацию	imgDelete	1	2	
Изменение шага жизненного цикла, дерево версий					
Lifecycle	Жизненный цикл	imgLCStepDocument	30	60	Keys.F5 Keys.Control
SetLifecycleStep	Изменить шаг жизненного цикла		10	10	
SetLifecycleStepChilds	Изменить шаг ЖЦ у объектов состава		10	15	
VersionHistory	История версии	10	20		
ObjectHistory	История объекта	10	30		
ListVersions	Версии объекта	imgVersionsTree	30	10	
ObjectComposition	Состав объекта	30	30		
Add	Добавить в состав	10	20		
Exclude	Исключить из состава	imgExclude	10	30	
CreateInComposition	Создать в составе	10	10	Keys.Delete	
BasedOnTemplate	Создать на основе шаблона	5	10		
Редактирование атрибутов					
Attributes	Атрибуты		30	70	
ShowAttributeHistory	История значений атрибута		5	10	
EditAttributeValue	Изменить значение атрибута		10	10	
AddAttribute	Добавить атрибут		10	20	

Название команды	Текстовое описание команды	Название изображения	Номер группы	Номер в группе	«Горячие» клавиши
AddAttributeGroup	Добавить группу атрибутов		10	30	
DeleteAttribute	Удалить атрибут		10	40	
DeleteAttributeGroup	Удалить группу атрибутов		10	50	
Просмотр, редактирование и печать объектов					
OpenDocument	Открыть		15	15	
OpenWith	Открыть с помощью...		15	20	
EditDocument	Редактировать		15	25	Keys.F4 Keys.Shift
ViewDocument	Смотреть	imgView	15	30	Keys.F3
PrintDocument	Вывести на печать	imgPrint	15	35	Keys.P Keys.Control
Специальные операции над объектами					
Synchronize	Синхронизировать		50	8	
Базовые операции над объектами					
CheckOut	Взять на редактирование	imgCheckOut	50	10	Keys.F9
CheckIn	Завершить редактирование	imgCheckIn	50	40	Keys.Control Keys.F10
SaveChanges	Сохранить изменения	imgSaveChanges	50	20	Keys.F10
CancelChanges	Отменить изменения	imgCancelChanges	50	30	Keys.F12
AdminCancelChanges	Отменить чужие изменения	imgAdminCancelChanges	50	50	Keys.C Keys.Shift Keys.Control
LocalizationCommand	Локализация		50	6	
Работа с составом объекта					
ConsistsFrom	Состоит из		58	20	Keys.F2 Keys.Control
EntersIn	Входит в		58	30	Keys.F2 Keys.Shift
Назначение системного GUID					
SetSystemGuid	Назначить системный Guid		80	1	
Операции с буфером обмена					
Cut	Вырезать	imgCut	60	10	Keys.X Keys.Control
Copy	Копировать	imgCopy	60	20	Keys.C Keys.Control
CopyText	Копировать текст	imgCopy	60	30	
Paste	Вставить	imgPaste	60	40	Keys.V Keys.Control

Название команды	Текстовое описание команды	Название изображения	Номер группы	Номер в группе	«Горячие» клавиши
Базовая версия, удаление объектов, объединение объектов, исключение из состава					
MakeBaseVersion	Сделать базовой версией		61	30	Keys.B Keys.Control
Delete	Удалить	imgDelete	62	40	Keys.Delete Keys.Control
CombineObjects	Объединить объекты		63	10	
Отметки					
Markers	Отметки	imgBookmark	70	10	
InvertMarkers	Инвертировать отметки	imgSelectionReplace	10	10	
MarkGroup	Отметить группу		10	20	
MarkGroupUp	Выше курсора		10	10	
MarkGroupDown	Ниже курсора		10	20	
MarkGroupAll	Все		10	30	
UnMarkGroup	Снять отметки у группы		10	30	
UnMarkGroupUp	Выше курсора		10	10	
UnMarkGroupDown	Ниже курсора		10	20	
UnMarkGroupAll	Все		10	30	
Действия с журналом событий					
CreateFilter	Создать фильтр	imgEventLogCreateFilterIcon	200	5	
DeleteEventLogRecord	Удалить запись	imgDelete	200	10	
ClearEventLog	Очистить журнал		200	20	
EventLogExcelReport	Отчет в Exel		200	20	
GetFileHistory	Сохранить файл		1000	0	
Доступ к страницам свойств					
Permissions	Доступ	imgAccess	10000	10	
Properties	Свойства	imgProp	10000	20	
Недавние объекты, настройка отображения кнопок, ручная сортировка					
RemoveRecentObjects	Исключить из списка		19990	5	
ClearRecentObjects	Очистить список		19990	10	
NavigatoContextSearch	Найти текст	imgFindText	19991	1	Keys.Control Keys.F
NavigatoContextSearchNext	Найти далее	imgFindTextNext	19991	2	Keys.Alt Keys.F
ManualSortingSetup	Ручная сортировка	imgManualSortingSetup	19992	9	
SetupColumns	Настройка отображения	imgViewSettings	20000	15	

Название команды	Текстовое описание команды	Название изображения	Номер группы	Номер в группе	«Горячие» клавиши
ResetColumns	Сбросить настройки отображения		20000	20	