



[www.klinkmann.com](http://www.klinkmann.com)

## Руководство пользователя ИСР Orchestra™

Редакция С

Дата пересмотра: 13.09.2005 г.

© 2006 Klinkmann. Все права защищены



[www.klinkmann.com](http://www.klinkmann.com)

Санкт-Петербург  
Москва  
Київ  
Tallinn  
Rīga  
Vilnius  
Helsinki

тел. +7 812 327 3752; [klinkmann@klinkmann.spb.ru](mailto:klinkmann@klinkmann.spb.ru)  
тел. +7 495 461 3623; [moscow@klinkmann.spb.ru](mailto:moscow@klinkmann.spb.ru)  
тел. +38 044 239 1250; [klinkmann@klinkmann.kiev.ua](mailto:klinkmann@klinkmann.kiev.ua)  
tel. + 372 668 4500; [klinkmann.est@klinkmann.ee](mailto:klinkmann.est@klinkmann.ee)  
tel. +371 738 1617; [klinkmann@klinkmann.lv](mailto:klinkmann@klinkmann.lv)  
tel. +370 5 215 1646; [post@klinkmann.lt](mailto:post@klinkmann.lt)  
ph. +358 9 540 4940; [automation@klinkmann.fi](mailto:automation@klinkmann.fi)

Все права зарезервированы. Дублирование, хранение в справочной системе, а также передача настоящего руководства, как целиком, так и частями, в любом виде (электронном, печатном, фотографическом и ином другом) без предварительного письменного согласия со стороны Invensys Systems, Inc. запрещается. Никакая ответственность по авторским правам и патентам в результате использования информации, содержащейся в настоящем документе, не возникает. Несмотря на то, что при подготовке настоящего руководства и соблюдались все нужные меры контроля, ни издатель, ни авторы не несут никакой ответственности за возможные ошибки или опечатки. Кроме того, не предполагается возникновения никакой ответственности за ущерб, причинённый вызовом информации, которая содержится в настоящем руководстве.

Информация, приведённая в настоящем руководстве, может модифицироваться и корректироваться без какого-либо предварительного уведомления и ни в какой мере не представляет собой какие-либо обязательства со стороны Invensys Systems, Inc. Описываемое в данном документе программное обеспечение поставляется в соответствии с условиями лицензии или соглашения о нераспространении. Указанное программное обеспечение может использоваться и копироваться только в соответствии с положениями данных документов.

**© 2002-2005 Invensys Systems, Inc. Все права защищены.**

Invensys Systems, Inc.  
26561 Rancho Parkway South  
Lake Forest, CA 92630 USA  
(949) 727-3200  
<http://www.wonderware.com>

### **Торговые марки**

Все используемые в настоящем руководстве термины, известные как торговые марки или служебные обозначения, выделены соответствующим образом. Invensys Systems, Inc. не имеет возможности проверить достоверность этой информации. Вызов какого-либо термина в настоящем руководстве не должно рассматриваться как подтверждение достоверности указанной торговой марки или служебного обозначения.

Аларм Logger, ActiveFactory, ArchestrA, Avantis, DBDump, DBLoad, DTAnalyst, FactoryFocus, FactoryOffice, FactorySuite, FactorySuite A<sup>2</sup>, InBatch, InControl, IndustrialRAD, IndustrialSQL Server, InTouch, InTrack, MaintenanceSuite, MuniSuite, QI Analyst, SCADAalarm, SCADASuite, SuiteLink, SuiteVoyager, WindowMaker, WindowViewer, Wonderware и Wonderware Logger являются торговыми знаками компании Invensys plc, её дочерних компаний и подразделений. Все остальные наименования могут представлять собой торговые марки соответствующих владельцев.

# Содержание

## Предварительные сведения ..... 7

Об этом руководстве .....	7
Принятые условные обозначения.....	8
Техническая поддержка .....	8

## ГЛАВА 1 Введение ..... 9

ИСП .....	9
Запуск ИСП .....	10
Соединение с Galaxy .....	10
Лицензирование .....	13
Регистрация в системе .....	15
Создание ярлыков Рабочего стола.....	16
Запуск нескольких экземпляров ИСП.....	17
Смена Galaxy.....	17
Интерфейс пользователя ИСП .....	18
Панель меню.....	19
Панель инструментов .....	23
Панель шаблонов .....	24
Структура приложения.....	26
Общая структура .....	26
Редактор объектов.....	30
Панель состояния .....	30
Панель операций .....	31
Настройка рабочей области.....	32
Комбинации клавиш.....	32
Допустимые имена и символы .....	33
Связь узел-узел .....	34
Учётные сведения о пользователях Archestra.....	35
Компьютеры с несколькими сетевыми адаптерами.....	35
Требования к минимальному дисковому пространству.....	40

## ГЛАВА 2 Объекты..... 41

Объекты .....	41
Импортирование объектов.....	42
Экспортирование объектов.....	45
Конфигурирование объектов .....	47
Захват и освобождение объектов.....	47
Удаление объектов.....	49
Переименование объектов.....	50
Расширение функциональных возможностей объектов .....	51
Шаблоны объектов .....	51
Управление шаблонами .....	51
Создание наборов шаблонов .....	52
Удаление наборов шаблонов.....	53
Создание шаблонов на основе других шаблонов.....	53
Блокирование и разблокирование атрибутов шаблона.....	54
Экземпляры объектов.....	55

Импортирование файлов определений объектов.....	56
Экспортирование определений объектов в файлы .....	57
Структура CSV-файла экспортирования .....	58
Создание экземпляров объектов по шаблонам .....	59
Значки состояний объектов.....	60
Проверка допустимости конфигурации объектов .....	61
Ручная проверка допустимости конфигурации.....	61
Построение приложения .....	62
Связь объектов с хостами.....	63
Связи по умолчанию.....	63
Пересылка объектов в узлы использования .....	65
Повторная пересылка объектов.....	68
Отмена использования объектов.....	69
Условия, возникающие в результате отмены использования.....	70

### **ГЛАВА 3 Редакторы объектов ..... 71**

Запуск и прекращение работы редактора объектов.....	71
Конфигурирование объекта в редакторе .....	72
Общий вид окна редактора объектов .....	72
Блокировка.....	74
Контроль доступа.....	74
Групповое блокирование и управление доступом .....	76
Функциональные клавиши .....	76
Просмотр объектов с помощью браузера атрибутов .....	76
Страница сведений об объекте .....	82

### **ГЛАВА 4 Архивирование ..... 85**

Архиватор InSQL.....	85
Настройка объектов WinPlatform и AppEngine на архивирование данных .....	86
Настройка остальных объектов для архивирования данных.....	86
Типы архивируемых атрибутов .....	88
Расширение набора архивируемых атрибутов .....	89
Изменение режима использования объекта и InSQL.....	89

### **ГЛАВА 5 Алармы и события ..... 91**

События и алармы .....	91
Система InTouch как получатель сведений об алармах и событиях .	92
Запрет алармов объекта AutomationObject .....	93
Разрешение алармов объекта AutomationObject .....	94
Конфигурирование объектов AutomationObject .....	94
Алармы, определяемые пользователем.....	95
Объекты, способные генерировать алармы и события .....	95
Объекты AutomationObject типа Atea .....	95
Подписка на получение сведения об алармах и событиях .....	96
Разрешение и блокирование алармов.....	96
Функционирование поставщика алармов при отключении сети. 97	
Различия между алармами и событиями в IAS и в InTouch .....	97

**ГЛАВА 6 Расширение функциональных возможностей объектов ..... 99**

Страницы расширений в окне редактора объектов .....	99
Наследование расширений.....	100
Скрипты .....	100
Страница Скрипты (Scripts) .....	101
Синхронное и асинхронное исполнение скриптов .....	106
Типы ошибок исполнения .....	107
Вызов функций в скриптах.....	107
Определяемые пользователем атрибуты .....	113
Страница Пользовательские атрибуты (UDAs).....	114
Расширения атрибутов .....	116
Страница Расширения (Extensions) .....	116
Расширенная характеристика в/в InputOutput .....	119
Расширенная характеристика ввода Input.....	120
Расширенная характеристика вывода Output .....	120
Расширенная характеристика алармов Alarm.....	121
Расширенная характеристика архивирования History .....	121
Выполнение операций вывода .....	122
Язык скриптов QuickScript .NET .....	123
Общие правила разработки скриптов.....	125
Меню правой кнопки в окне создания скриптов.....	129
Скриптовые функции.....	131
Переменные QuickScript .NET .....	161
Числовые и символьные значения.....	162
Управляющие структуры языка QuickScript .NET .....	164
Распознавание ключевых слов и идентификаторов .....	167
Примеры скриптов .....	167

**ГЛАВА 7 Объекты-контейнеры ..... 179**

Вложенные объекты .....	179
Примеры объектов-контейнеров.....	179
Изменение вложенного имени объекта.....	180
Контейнеры ApplicationObject.....	181

**ГЛАВА 8 Ссылки ..... 183**

Обмен сообщениями.....	183
Виды ссылок.....	183
Формат ссылок .....	184
Ссылки и перекрёстные ссылки .....	187
Окно свойств объекта.....	187
Окно поиска объектов .....	191
Просмотр тэгов и ссылок Galaxy из приложений InTouch.....	193

**ГЛАВА 9 Контроль доступа ..... 197**

Определение прав доступа.....	198
Способ идентификации пользователей.....	199
Группы прав доступа .....	202
Роли .....	203
После изменения прав доступа.....	209

Предоставление доступа в режиме проверки полномочий групп ОС209

## **ГЛАВА 10 Управление Репозиторием Galaxy211**

Определение состояния Galaxy .....	211
Создание новой Galaxy .....	212
Наличие нескольких Galaxy в одном Репозитории Galaxy .....	213
Загрузка и выгрузка Galaxy .....	213
Выгрузка рабочих параметров в Galaxy .....	213
Резервное копирование и восстановление Galaxy .....	214
Конфигурирование главного источника показаний времени .....	214

## **ГЛАВА 11 Резервирование компонентов Arcestra..... 217**

Общие сведения.....	217
Термины.....	218
Примеры конфигураций с резервированием .....	219
Резервирование объектов AppEngine.....	220
Конфигурирование.....	221
Значки структуры приложения .....	224
Команды ИСР .....	224
Ошибки и предупреждения.....	226
Рассылка объектов AppEngine .....	226
Действия, выполняемые системой в случае отказа одного из компонентов .....	228
Генерация архивной информации .....	232
Принудительное переключение.....	232
Резервирование каналов связи.....	232
Конфигурирование резервных источников данных.....	232
Пересылка.....	233
Переключение источников данных в рабочем приложении .....	233

## **Словарь терминов Arcestra..... 235**

# Предварительные сведения

## Об этом руководстве

Настоящее руководство описывает интерфейс пользователя и функции ИСП – ИСП (Integrated Development Environment – IDE) ArchestrA™. Эта книга организована следующим образом:

- "Введение": детальное описание элементов пользовательского интерфейса ИСП.
- "Работа с объектами": описание импортирования и работы с шаблонами объектов, работы с наборами инструментов в **Template Toolbox (панель шаблонов)** ИСП, а также создания, конфигурирования и внедрения экземпляров объектов.
- "Работа с редакторами объектов": описание использования редакторов конфигурирования объектов.
- "Архивирование": описание взаимосвязей между объектами Industrial Application Server и функциями архивирования, предоставляемыми InSQL.
- "Алармы и события": описание событий и алармов (тревог), различий между ними, а также описание того, как конфигурировать объекты для отчётности через провайдер алармов InTouch.
- "Расширение функциональности объектов": описание использования страниц **скриптов, пользовательских атрибутов – UDA (User-Defined Attribute)** и **расширений** редактора объектов для расширения функциональности по сравнению с их первоначальными возможностями.
- "Работа с контейнерами": описание использования объектов встраивания для построения сложных устройств практического применения.
- "Работа со ссылками": описание использования строк ссылок, диалогового окна **Properties (свойства)** и поиска потерянных объектов.
- "Работа с защитой данных": описание конфигурирования прав доступа в среде ArchestrA.
- "Управление Galaxy": описание базы данных системы SQL Server для Репозитория Galaxy, резервного копирования Galaxy, а создания, удаления и конфигурирования главного узла времени Galaxy.
- "Резервирование ArchestrA": описание функций резервного копирования в инфраструктуре ArchestrA для предоставления стратегического дублирования критических компонентов вашего приложения.
- "Предметный указатель".

Подробнее об ArchestrA см. в *Руководстве по разработке проектов FactorySuite A<sup>2</sup>*.

Вы можете просмотреть этот документ в электронном виде, а также распечатать его, целиком или полностью, используя средства печати Adobe Acrobat Reader.

## Принятые условные обозначения

В настоящем документе действуют следующие соглашения:

Условное обозначение	Назначение
Полужирный шрифт	Меню, команды, кнопки, значки, диалоговые окна и параметры диалоговых окон.
Моноширинный шрифт	Начальные выделения меню, текст, который вы должны ввести и программный код.
<i>Курсив</i>	Параметры в тексте или программном коде, которые вы должны ввести.

## Техническая поддержка

Служба технической поддержки Wonderware обеспечивает широкие возможности для ответа на любые вопросы по продуктам Wonderware и их применению.

Прежде чем связываться со службой технической поддержки, обращайтесь, пожалуйста, к соответствующим разделам вашего *Руководства пользователя ИСП Orchestra* для возможного решения проблем, которые вы можете встретить при использовании ИСП. Если вы сочтёте необходимым связаться для консультаций со службой технической поддержки, пожалуйста, имейте доступной следующую информацию:

- Тип и версию операционной системы, которую вы используете. Например Microsoft Windows XP.
- Точную формулировку полученного сообщения об ошибке.
- Любые относящиеся к делу выходные листинги Log Viewer (Обозреватель журналов) или других диагностических программ.
- детальное описание попыток, которые вы предприняли для устранения проблем(ы), и полученных результатов.
- детальное описание, как воспроизвести проблему.
- Номер, если он известен, присвоенный вашей проблеме службой технической поддержки Wonderware (если это повторная проблема).



## ГЛАВА 1

# Введение

ArchestrA™ представляет собой архитектуру следующего поколения для разработки производственных информационных систем и систем диспетчерского управления. Она является открытой расширяемой системой компонентов базирующейся на распределённом объектно-ориентированном подходе. Настоящее руководство содержит сведения об использовании основного инструментального средства ArchestrA – ИСР (ИСР).

В настоящей главе описаны способы запуска ИСР и основные компоненты пользовательского интерфейса.

Подробнее об ArchestrA см. в Руководстве по разработке проектов FactorySuite A<sup>2</sup>™.

## Содержание

- ИСР
- Запуск ИСР
- Смена Galaxy
- Интерфейс пользователя ИСР
- Комбинации клавиш
- Допустимые имена и символы
- Связь узел-узел
- Требования к минимальному дисковому пространству

## ИСР

ИСР ArchestrA представляет собой интегрированную инструментальную систему проектирования и разработки, с помощью которой конфигурируются объекты ArchestrA и передаются на целевые компьютеры. Эти система позволяет настраивать параметры объектов, составляющих прикладную систему, и инфраструктуры, лежащей в её основе.

С помощью ИСР можно импортировать в Galaxy новые типы объектов (шаблоны), создавать на их основе новые объекты и экспортировать их в различные компьютеры общей сети. В одной и той же Galaxy допускается одновременная работа с различными наборами объектов из разных ИСР.

ИСР может устанавливаться на любом компьютере, на котором имеется программа загрузки ArchestrA Bootstrap.

## Запуск ИСР

Чтобы запустить ИСР, щёлкните **Пуск (Start)** Панели задач Windows, расположите курсор мыши последовательно на **Программы Programs)** и **Wonderware** и щёлкните **ИСР ArchestrA (ArchestrA IDE)**.

Для появления ИСР нужно выполнение следующих условий:

- В Репозитории Galaxy должна быть определена хотя бы одна Galaxy;
- Существует правильная лицензия вашего Репозитория Galaxy.

Также может потребоваться ввод учётной информации, если для указанной Galaxy включен контроль доступа. Система защиты информации ArchestrA определяет, к каким компонентам ИСР и утилитам системной консоли управления (System Management Console) и среды исполнения имеет доступ тот или иной пользователь. При отказе доступа к ИСР или её стандартным функциям обратитесь к системному администратору. Подробнее см. главу "Контроль доступа" настоящего руководства.

**Примечание.** В разных системах Microsoft® Windows® некоторые компоненты пользовательского интерфейса могут различаться по внешнему виду и исполнению соответствующих функций. Настоящее руководство отражает работу в Windows XP.

## Соединение с Galaxy

Подключение Galaxy обязательно для любых действий в ИСР ArchestrA. Запустить ИСР без подключения к какой-либо из систем ArchestrA невозможно. При запуске ИСР появляется окно **Соединение с Galaxy (Connect to Galaxy)**:




Окно содержит следующие три группы элементов:

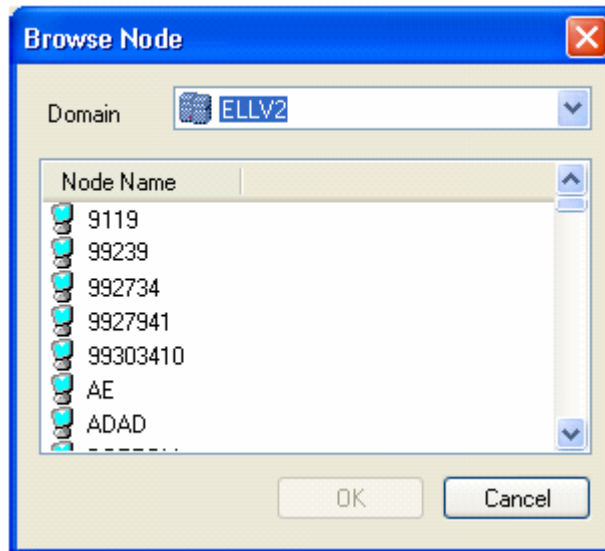
- Выбор подключения, состоящий из полей **Имя узла Репозитория Galaxy (GR Node Name)** и **Имя Galaxy (Galaxy Name)**.
- Кнопки действий: **Соединиться (Connect)**, **Создать Galaxy (New Galaxy)**, **Удалить Galaxy (Delete Galaxy)**, **О программе (About)**, **Отмена (Cancel)**.
- Лицензионная информация.

Если список поля **Имя Galaxy (Galaxy Name)** пуст, значит на компьютере с именем, указанным в поле **Имя узла Репозитория Galaxy GR Node Name)**, ещё не создана ни одна Galaxy. Чтобы начать сеанс работы в ИСР

ArchestrA, нужно выбрать существующую Galaxy на другом узле или создать новую на данном узле.

### Чтобы просмотреть другие узлы

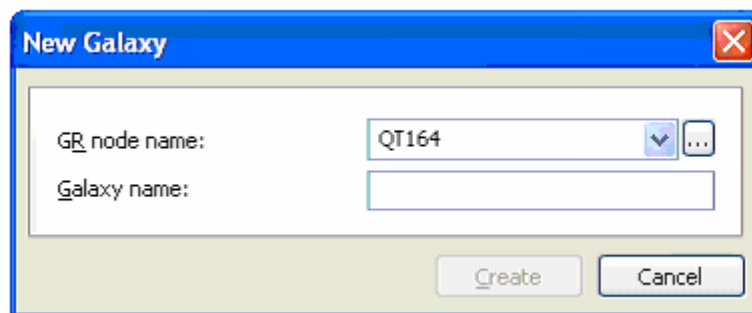
1. Щёлкните в диалоге **Подключиться к Galaxy (Connect to Galaxy)** кнопку просмотра . Появится окно **Поиск узла (Browse Node)**.



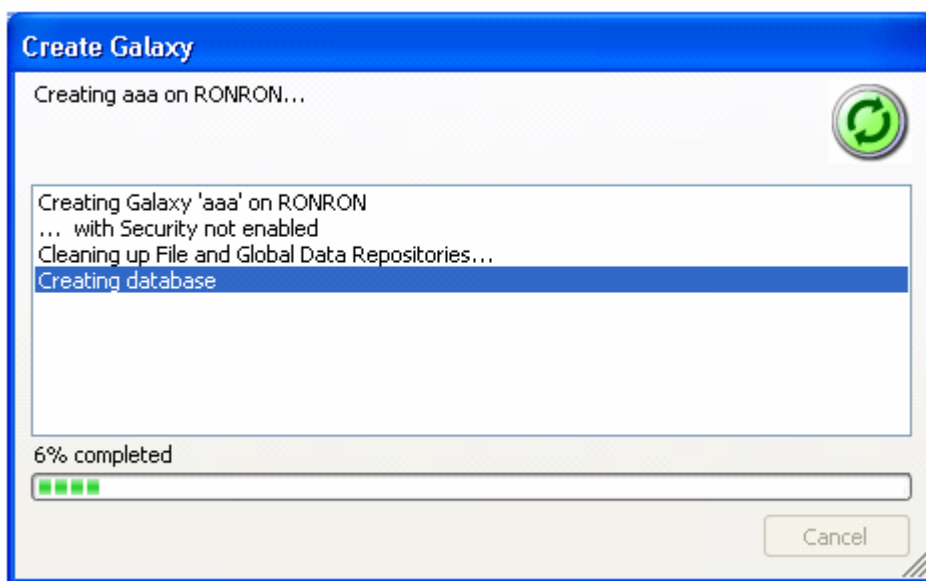
2. Выберите из списка **Домен (Domain)** нужный домен сети.
3. Выделите в поле **Имя узла (Node Name)** требуемое имя компьютера.
4. Щёлкните **ОК**, чтобы подтвердить выбор, или **Отмена (Cancel)**, чтобы отменить выбор и возвратиться в окно соединения с Galaxy.

### Чтобы создать новую Galaxy

1. В окне **Соединение с Galaxy (Connect to Galaxy)** щёлкните **Новая Galaxy (New Galaxy)**. Появится окно **Новая Galaxy (New Galaxy)**.



2. Выберите из списка **Имя узла Репозитория Galaxy (GR Node Name)** компьютер, на котором должна быть создана новая Galaxy.
3. Введите в поле **Имя Galaxy (Galaxy Name)** имя новой Galaxy.
4. Щёлкните **ОК**. Появится индикатор хода создания новой системы. Чтобы возвратиться в окно **Соединиться с Galaxy (Connect to Galaxy)**, не создавая новую систему, щёлкните **Отмена (Cancel)**.



---

**Примечание.** В только что созданной Galaxy параметры защиты информации не определены. Кроме того, новые системы имеют следующие характеристики: количество пользователей – 2 (DefaultUser и Administrator, имеющие полный доступ ко всем функциям системы), количество ролей – 2 (Default и Administrator, с полным набором полномочий), количество групп доступа – 1 (Default). Подробнее о модели защиты данных и установке параметров контроля доступа в новой системе см. главу "Контроль доступа" настоящего руководства.

---

Для возврата в окно **Соединиться с Galaxy (Connect to Galaxy)** щёлкните **Заккрыть (Close)**. После выбора требуемых имён узлов Репозитория Galaxy и системы Galaxy щёлкните **Соединиться (Connect)**, чтобы запустить ИСП ArchestrA и подключиться к указанной Galaxy.

Если в указанном узле Репозитория существует только одна Galaxy, выбор её имени будет сделан автоматически. Щёлкните **Соединиться (Connect)**, чтобы запустить ИСП ArchestrA и подключиться к этой системе. Если в узле существует несколько Galaxy, выбрана будет последняя система, с которой когда-либо устанавливалось соединение. Выберите из списка **Имя Galaxy (Galaxy Name)** нужную систему и щёлкните **Соединиться (Connect)**.

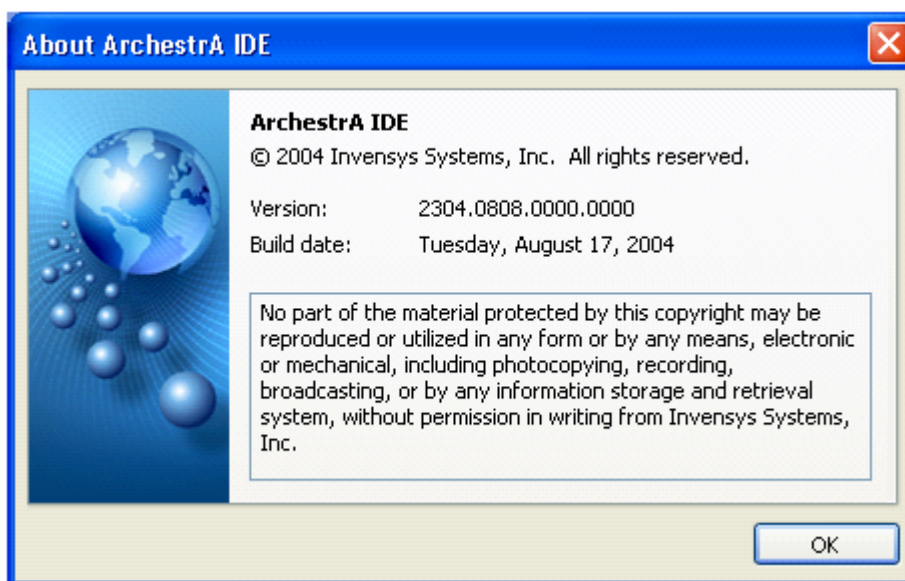
---

**Примечание.** Если параметры контроля доступа будут установлены, сначала откроется окно **Регистрация (Login)**. Если контроль доступа будет отключен, пользователь автоматически будет зарегистрирован в системе как пользователь "DefaultUser". Подробнее о регистрации см. параграф "Регистрация в системе".

---

Чтобы удалить Galaxy, имя которой выделено в списке **Имя Galaxy (Galaxy Name)**, щёлкните **Удалить Galaxy (Delete Galaxy)**.

Чтобы определить версию программного обеспечения, авторские права и другую информацию, щёлкните **О программе (About)**. Появится окно **Об ИСП ArchestrA (About ArchestrA IDE)**. Чтобы отменить запуск ИСП, щёлкните **Отмена (Cancel)**. В качестве примера на следующем рисунке показано окно выводимых сведений.



В панели **Клиентская лицензия Galaxy (Galaxy Client Access License)** отображается следующая информация:

- Номер лицензии.
- Наименование поставщика.
- Описание системы.
- Дата окончания действия (если таковая определена).
- Примечания.

## Лицензирование

Доступ к Репозиторию Galaxy определяется на основании лицензии. Если при попытке запустить ИСР появляется сообщение, в котором содержится ссылка на лицензию, скорее всего, лицензионная информация недействительна. Указанное сообщение может выводиться в следующих случаях:

- На компьютере нет лицензий.
- Срок действия лицензии истёк
- Превышено количество точек в/в или количество WinPlatform.

Устранить эти проблемы можно с помощью утилиты License Utility. До их устранения будут невозможны следующие действия:


- Запуск ИСР.
- Подключение к существующим Galaxy.
- Создание новых Galaxy.

После обновления лицензии подключение к Galaxy и запуск ИСР будут выполняться без проблем.

---

**Примечание.** Если срок действия лицензии заканчивается во время сеанса работы с ИСР, при очередном открытии окна ИСР подключиться к Galaxy не удастся.

---

Чтобы проверить допустимость лицензии, срок её действия и существующие ограничения, щёлкните дважды значок лицензии  в нижней части главного окна ИСР. Появится окно **Информация о лицензии (License Information)**.

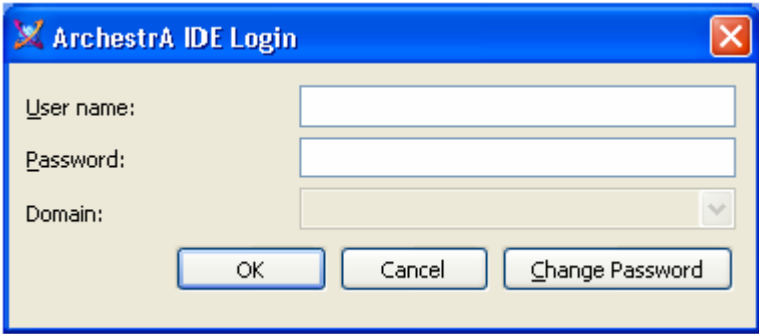
В данное окно выводятся следующие сведения:

- Группа **Клиентская лицензия Galaxy (Galaxy Client Access License)**:
  - **Номер (Number)**: номер лицензии;
  - **Поставщик (Vendor)**: обозначение поставщика программного обеспечения;
  - **Описание продукта (Product Text)**: имя программы и номер версии;
  - **Срок действия (Expiry Date)**: срок окончания действия лицензии;
  - **Текст замечания (Notice Text)**: дополнительная информация о поставщике программного обеспечения.
- Группа **Платформа (Platform)**:
  - **Текущее количество (Current Count)**: текущее количество WinPlatform в Galaxy;
  - **Макс. количество (Max Count)**: максимально допускаемое лицензией количество WinPlatform. Если количество точек в/в и платформ в лицензии не указано, в данном поле отображается текст "N/A";
  - **Статус (Status)**: связь между текущим и максимальным значениями количества платформ. Возможны три типа связи: OK, Exceeded (количество используемых платформ превышает максимально допустимое) и DEV (количество точек в/в и WinPlatform не определено).
- Группа **Точки в/в (IO Point)**:
  - **Количество сконфигурированных (Configured Count)**: количество сконфигурированных в Galaxy точек в/в;

- **Макс. количество (Max Count):** максимально допускаемое лицензией количество точек в/в. Если количество точек в/в и платформ в лицензии не определено, в данном поле отображается текст "N/A";
- **Статус (Status):** связь между текущим и максимальным значениями количества точек в/в. Возможны три типа связи: OK, Exceeded (количество используемых точек в/в превышает максимально допустимое) и DEV (в лицензии количество точек в/в и WinPlatform не определено);

## Регистрация в системе

При установлении соединения с Galaxu, в которой осуществляется контроль доступа, появится окно **Регистрация в ИСП ArchestrA (ArchestrA IDE Login)**:

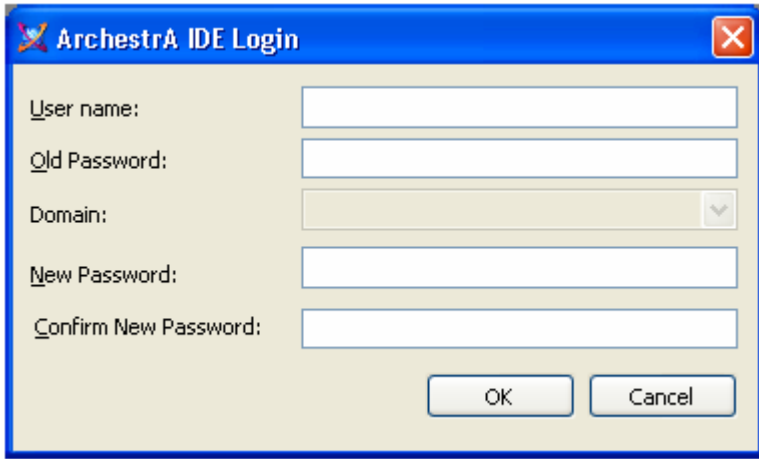


Введите в поля **Имя пользователя (User Name)** и **Пароль (Password)** соответственно имя пользователя и пароль доступа, который отображается при вводе в виде звёздочек. Если в Galaxu осуществляется проверка полномочий на уровне операционной системы, дополнительно нужно указать домен, для которого действительна введённая учётная информация, при условии, что это не локальный домен.

Щёлкните **OK**. Система проверит допустимость введённых сведений для доступа в указанный Репозиторий Galaxu (и операционную систему) и запустит ИСП ArchestrA. Если введённое имя пользователя или пароль в Репозитории Galaxu не определён, появится соответствующее сообщение.

Для отмены регистрации и завершения сеанса работы с ИСП щёлкните **Отмена (Cancel)** или клавишу ESC.

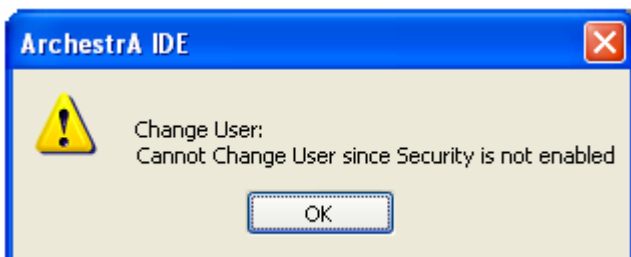
Чтобы изменить пароль доступа, щёлкните **Изменить пароль (Change Password)**. Эта кнопка будет недоступна до тех, пока пользователь не введёт допустимое имя и пароль доступа.



Подробнее см. главу "Контроль доступа" настоящего руководства.

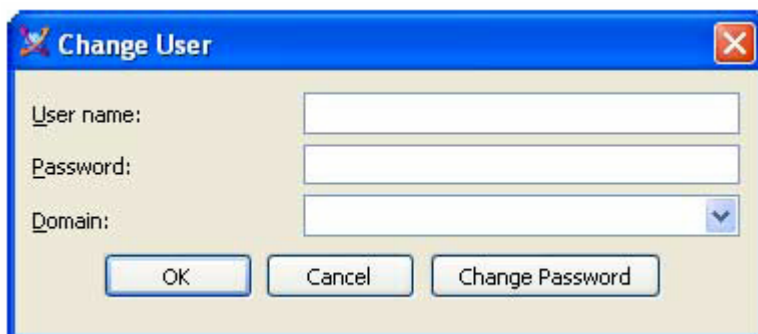
## Смена пользователей

В любой момент во время работы с ИСП может быть выполнена смена пользователя с помощью команды **Сменить пользователя (Change User)** из меню Galaxy. Если функция контроля доступа не включена, появится следующее сообщение:



Щёлкните **ОК**. Всем пользователям в Galaxy с отключенным контролем доступа предоставляются пользовательские права группы Defaultuser (Пользователь по умолчанию).

Если функция контроля доступа включена, появится окно **Смена пользователя (Change User)**:



Введите учётные данные нового пользователя и щёлкните **ОК**. Если переключение пользователей выполнять не нужно, щёлкните **Отмена (Cancel)**. Чтобы изменить пароль нового пользователя, щёлкните **Изменить пароль (Change Password)**. Появится следующее окно:



## Создание ярлыков Рабочего стола

Упростить подключение к Galaxy можно с помощью ярлыков Рабочего стола.

Чтобы создать на Рабочем столе ярлык:



1. Щёлкните правой кнопкой мыши в любой точке Рабочего стола, поместите указатель на строку **Новый (New)** и щёлкните левой кнопкой пункт **Ярлык (Shortcut)**.
2. В появившемся окне **Создание ярлыка (Create Shortcut)** щёлкните **Обзор (Browse)**.
3. Перейдите в каталог установки ИСР (aalde.exe).
4. Выделите aalde.exe и щёлкните **Открыть (Open)**.
5. Щёлкните **Далее (Next)**.
6. Введите текст для названия ярлыка и щёлкните **Завершить (Finish)**.
7. Щёлкните правой кнопкой мыши значок созданного ярлыка и выполните в появившемся меню команду **Свойства (Properties)**.
8. Перейдите на страницу **Ярлык (Shortcut)** и установите курсор в конец поля **Объект (Target)**. Введите пробел, имя узла Репозитория Galaxy, обратную наклонную черту ("\"") и имя Galaxy.
9. Щёлкните **ОК**.

Для установления соединения дважды щёлкните созданный ярлык. Если параметры ярлыка указаны неправильно, появится соответствующее сообщение об ошибке. Щёлкните **ОК** – появится окно **Соединение с Galaxy (Connect to Galaxy)**. Если сведения о Galaxy в ярлыке указаны неверно, в поле **Имя Galaxy (Galaxy Name)** будет показано имя последней системы, с которой устанавливалось соединение.

---

**Внимание!** Учётные сведения в ярлыке Рабочего стола указать невозможно. Если в Galaxy включена функция контроля доступа, после двойного щелчка ярлыка появится окно **Регистрация (Login)**.

---

## Запуск нескольких экземпляров ИСР

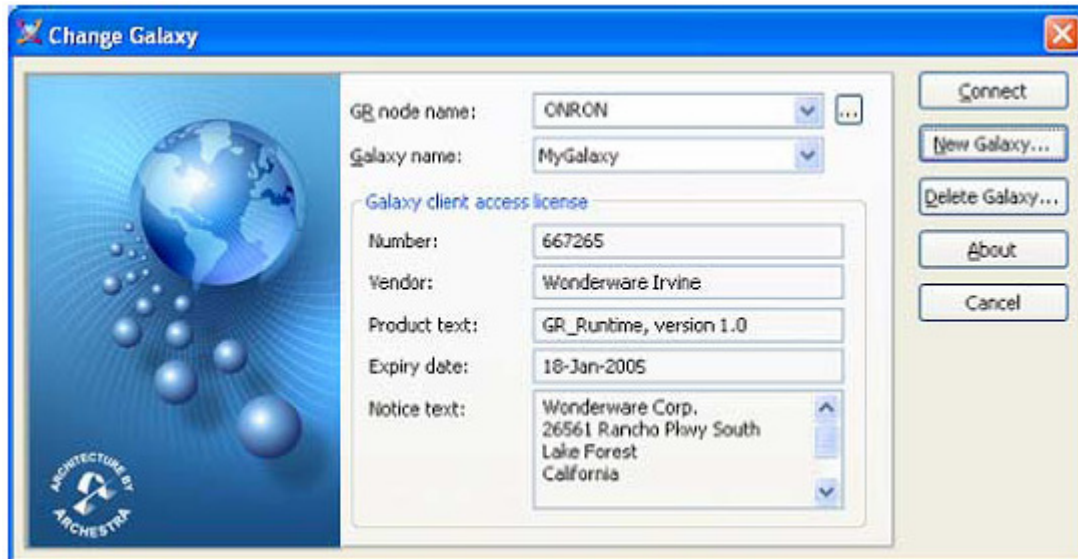
Соединиться с конкретной Galaxy и определить её параметры могут одновременно несколько пользователей. При этом каждый из них запускает ИСР и подключается к Galaxy обычным образом.

ArchestrA гарантирует:

- Что объекты, выбранные одним пользователем, другой пользователь редактировать не сможет. Но любой пользователь может определить, кто именно "занял" тот или иной объект Galaxy;
- Что изменения конфигурации, выполненные каким-либо одним пользователем, отображаются в ИСР всех других пользователей.

## Смена Galaxy

Чтобы перейти в другую Galaxy, нужно выполнить команду меню **Сменить Galaxy (Change Galaxy)**. Появится окно **Сменить Galaxy (Change Galaxy)**.

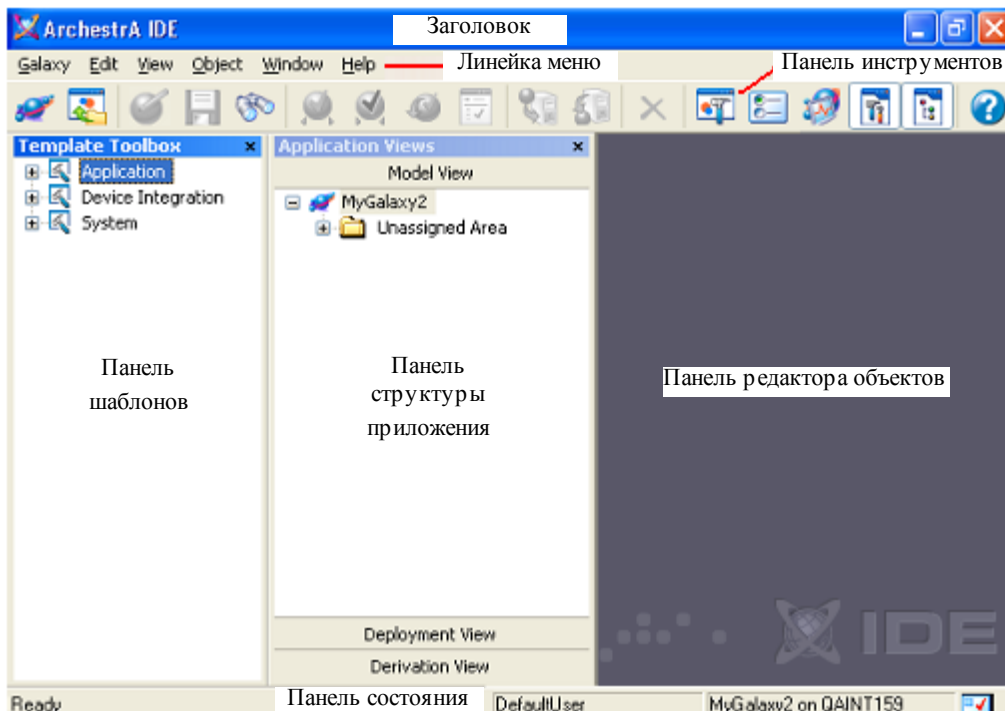


Это окно позволяет выбрать другую Galaxy (поле **Имя Galaxy – Galaxy Name**) или другой Репозиторий Galaxy (поле **Имя узла Репозитория Galaxy – GR Node Name**), создать новую Galaxy (кнопка **Новая Galaxy – New Galaxy**) или удалить существующую (кнопка **Удалить Galaxy – Delete Galaxy**). Чтобы сменить систему, выберите нужные названия в полях **Имя узла Репозитория Galaxy (GR Node Name)** и **Имя Galaxy (Galaxy Name)** и щёлкните **Connect (Соединиться)**.

Другие поля такие же, как и у окна **Соединение с Galaxy (Connect to Galaxy)**.

## Интерфейс пользователя ИСР

Основным интерфейсным элементом ИСР является главное окно, в котором осуществляется разработки приложений и рассылка их по домену.



Главное окно ИСР содержит следующие элементы:

- Заголовка.

- Панели меню.
- Панели инструментов.
- Панели шаблонов.
- Панели структуры приложения.
- Панели редактора объектов.
- Панели состояния.

Эти элементы отображаются по умолчанию. В состав главного окна входит также панель выполняемых операций, которая изначально не отображается.

В заголовок выводится название утилиты.

Остальные компоненты главного окна описаны в следующих параграфах.

## Панель меню

Панель меню представляет собой один из динамических элементов главного окна, состоящий из следующих меню: **Galaxy**, **Редактировать (Edit)**, **Вид (View)**, **Объект (Object)**, **Окно (Window)** и **Справка (Help)**.

Доступность тех или иных команд меню определяется тем, какой из объектов или элементов главного окна выделен в текущий момент, в каком он состоянии, какие функции логически разрешены для него и т.д. Далее приведено краткое описание всех команд меню:

Меню **Galaxy**: содержит общесистемные и глобальные команды пользовательского уровня.

- **Создать (New)**
  - **Экземпляр (Instance)**: создание объекта на основе выделенного в текущий момент шаблона.
  - **Производный шаблон (Derived Template)**: создание нового шаблона на основе выделенного;
  - **Набор шаблонов (Template Toolset)**: создание нового набора шаблонов. При выполнении данной команды появляется окно **Новый набор (New Toolset)**.
- **Открыть (Open)**: запуск редактора выделенного объекта.
- **Открыть только для чтения (Open Read-Only)**: запуск редактора выделенного объекта в режиме "только для чтения". Эта команда может выполняться в следующих условиях: когда объект отмечен другим пользователем; когда у пользователя нет соответствующих полномочий в целом или полномочий на модификацию выделенного объекта и т.д.
- **Закреть (Close)**: закрытие окна редактора. При этом будет открыто окно с запросом сохранения всех сделанных изменений.
- **Импортирование (Import)**
  - **Объект(ы) автоматизации (Automation Object(s))**: импортирование файла определений объектов (с расширением .aardf и .aarkg). При выполнении данной команды появляется окно **Открыть (Open)**, в котором можно выбрать один или несколько файлов. Подробнее см. также параграф "Импортирование объектов" настоящего руководства.
  - **Библиотека скриптов (Script Function Library)**: импортирование в Galaxy библиотеки функций, то есть файлов с расширением .aaSLIB, COM.dll, COM.tlb, COM.olb или .NET.dll. Эти библиотеки содержат функции, которые могут быть

использованы внутри скриптов. Если имя импортируемой функции совпадает с уже существующим, система предложит заменить его.

- **Загрузка Galaxy (Galaxy Load):** загрузка конфигурационных параметров Galaxy из файла в формате CSV (comma separated values – разделённые запятыми значения), в который ранее были экспортированы системные объекты. (Подробнее см. описание команды **Данные дампа Galaxy – Galaxy Dump**). При выполнении данной команды появляется стандартное окно **Открыть (Open)** поиска файла.
- **Экспортирование (Export)**
  - **Объект(ы) автоматизации (Automation Object(s)):** экспортирование выделенных объектов в пакетный файл с расширением .aaPKG. При выполнении данной команды появляется стандартное окно **Сохранить как... (Save As)**. В указанный файл копируются атрибуты и параметры исходных объектов. Подробнее см. параграф "Экспортирование объектов" настоящего руководства.
  - **Все объекты автоматизации (All Automation Objects):** экспортирование всех объектов Galaxy в указанный пакетный файл с расширением .aaPKG.
  - **Библиотеки скриптов (Script Function Library):** экспортирование библиотек функций, которые могут использоваться внутри других скриптов. При выполнении данной команды появляется окно **Экспортирование библиотеки скриптов (Export Script Function Library)**.
  - **Данные дампа Galaxy (Galaxy Dump):** сохранение конфигурационных параметров Galaxy в файле в формате CSV. Экспортированный файл можно редактировать в любом приложении, поддерживающем этот формат, например в MS Excel. Сохранённую информацию можно загрузить в другую Galaxy с помощью команды **Загрузить Galaxy (Galaxy Load)**. При выполнении данной команды появляется стандартное окно **Сохранить (Save)**, в котором задаётся каталог и имя требуемого файла.
- **Сохранить (Save):** сохранение всех конфигурационных параметров, заданных в текущий момент в окне редактора.
- **Сохранить всё (Save All):** сохранение информации из всех открытых окон редактора.
- **Конфигурировать (Configure)**
  - **Права доступа (Security):** при выполнении данной команды появляется окно **Параметры доступа (Configure Security)**, в котором устанавливаются все параметры доступа ArchestrA. Подробнее см. главу "Контроль доступа" настоящего руководства.
  - **Проверка прав доступа (View Security):** при выполнении данной команды появляется окно **Параметры доступа (Configure Security)** в режиме "только для чтения" (изменять значения в этом окне нельзя).
  - **Источник времени (Time Master):** при выполнении данной команды появляется окно **Параметры источника времени (Configure Time Master)** для установки параметров сетевого узла, который будет выступать в роли источника эталонного времени.

- **Настройка наборов инструментов (Customize Toolsets):** при выполнении данной команды появляется окно **Customize Toolsets**, в котором указывается группы шаблонов, которые должны отображаться на панели шаблонов.
- **Статус Galaxy (Galaxy Status):** при выполнении данной команды появляется окно, в которое выводятся значения счётчиков, характеризующих состояние объектов Galaxy.
- **Свойства (Properties):** при выполнении данной команды появляется окно, в которое выводятся значения свойств выделенного объекта, его атрибутов и ссылок на него. Кроме того, в этом окне можно изменить параметры регистрации, предельные значения и тексты сообщений об ошибках и предупреждений.
- **Сменить Galaxy (Change Galaxy):** при выполнении данной команды появляется окно, в котором можно указать систему, отличную от той, в которой в данный момент работает пользователь.
- **Смена пользователя (Change User):** смена в текущем сеансе зарегистрированного пользователя ИСР. При выполнении данной команды появляется окно **Смена пользователя (Change User)**. Если функция контроля доступа не включена, будет выведено сообщение о том, что контроль доступа не осуществляется.
- **<Список Galaxy>:** список систем, с которыми осуществлялось соединение. Список состоит из четырёх элементов, причём на первом месте указана система, соединение с которой было самым недавним по времени. При выполнении данной команды, то есть щелчке кнопкой мыши на каком-либо элементе списка, появляется окно **Регистрация в ИСР ArchestrA (ArchestrA IDE Login)**, при условии, что функция контроля доступа включена.
- **Выход (Exit):** команда завершения сеанса работы с ИСР. При выполнении данной команды появляется окно с предложением сохранить информацию из открытых окон редактора. Пользовательские предпочтения, например расположение панели структуры приложения, для зарегистрированного пользователя сохраняются.

Меню **Редактировать (Edit)**. В этом меню собраны команды редактирования объектов.

- **Переименовать (Rename):** команда переименования выделенного объекта.
- **Изменить содержащееся имя (Rename Contained Name):** при выполнении данной команды появляется окно **Изменить содержащееся имя (Rename Contained Name)**, в котором можно изменить имя выделенного вложенного объекта.
- **Удалить (Delete):** удаление выделенного объекта.
- **Найти (Find):** при выполнении данной команды появляется окно, в котором настраиваются параметры поиска объектов по имени и типу.
- **Пользовательские настройки (User Information):** установка глобальных настроек зарегистрированного пользователя (например системные объекты по умолчанию). При выполнении данной команды появляется окно определения пользовательских настроек.

Меню **Вид (View)**. В данное меню входят команды, управляющие отображением тех или иных элементов главного окна ИСР. В первом сеансе работы пользователя с ИСР отображаются все три описываемых ниже элемента окна. В последующих сеансах начальная конфигурация окна восстанавливается из настроек, сохранённых пользователем во время предыдущего сеанса.

- **Представления приложения (Application Views):** команда фокусирования на некоторой части представлений приложения Основного окна (Main Window).
- **Панель шаблонов (Template Toolbox):** команда выделения панели шаблонов.
- **Панель операций (Operations):** команда отображения панели текущих операций. Чтобы убрать её, щёлкните символ "X".
- **Панель состояния (Status Bar):** команда показа и скрытия панели состояния.

Меню **Объект (Object)**. В данное меню входят команды, выполняющие определённые действия с объектами.

- **Захватить (Check Out):** команда захвата объекта Galaxy для монопольного изменения его параметров. Пока объект не будет освобождён, никакой другой пользователь не сможет изменить его характеристики.
- **Освободить (Check In):** команда освобождения ранее захваченного объекта. При выполнении данной команды появляется окно **Освободить объект (Check In Object)**.
- **Отменить захват (Undo Check Out):** отмена предшествующей команды захвата объекта без изменения его конфигурационных параметров и записи в журнал модификаций. После выполнения данной команды захватить объект может любой пользователь Galaxy.
- **Игнорировать захват (Override Check Out):** команда для игнорирования состояния флажка захвата объекта. Для выполнения данной команды требуется, как правило, наличие особых полномочий. Она должна использоваться тогда, когда конфигурация объекта не изменяется захватившим его пользователем.
- **Проверить (Validate):** команда проверки допустимости конфигурации одного или нескольких объектов. При её выполнении появляется панель **Операции (Operations)**. Подробнее см. параграф "Проверка конфигурации объектов".
- **Просмотр в Object Viewer (View in Object Viewer):** команда запуска утилиты Object Viewer, с помощью которой можно просмотреть значения атрибутов выделенного объекта. Данная команда доступна, только если выделенный объект используется.
- **Переслать (Deploy):** команда пересылки одного или нескольких выделенных объектов в узлы, определяемые конфигурационными параметрами этих объектов. При выполнении данной команды появляется окно **Рассылка объектов (Deploy Object)**.
- **Отменить использование объекта (Undeploy):** команда отмены одного или нескольких объектов, используемых в различных узлах системы. При выполнении данной команды появляется окно **Отменить использование объектов (Undeploy Object)**.
- **Привязать (Assign To):** при выполнении данной команды появляется окно **Assign To**, в котором определяется объект, к которому должен быть привязан выделенный объект.
- **Отменить привязку (Unassign):** команда отмены привязки выделенного объекта и помещения его в папку **Нераспределённые хосты (Unassigned Host)** Структуры использования (Deployment View) или в папку **Нераспределённые объекты (Unassigned Area)** Общей структуры (Model View). При выполнении данной команды в окне Структуры вывода (Derivation View) отменяются все родительские связи (хост, зона, вложение). Ход процесса отмены привязки

отображается в окне **Отмена привязки (Unassign)**. Если выделенный объект используется в каком-либо узле, команда не выполняется.

- **Установить как умолчание (Set as Default)**: команда определения системного объекта WinPlatform, AppEngine или Area в качестве объекта привязки других объектов по умолчанию.
- **Загрузка рабочих характеристик (Upload Runtime Changes)**: команда пересылки конфигурационных параметров выделенного используемого объекта в Galaxy. Эта команда обычно используется тогда, когда после изменения некоторых атрибутов объекта в среде конфигурирования окажется, что предпочтительными являются его рабочие характеристики. Использование данной команды позволяет заменить значения конфигурационных параметров в Galaxy рабочими параметрами.

Меню **Окно (Window)**. В данное меню входят команды манипулирования параметрами окна редактора объектов. Меню доступно, если открыто окно хотя бы одного редактора.


- **Каскад (Cascade)**: стандартная команда Windows для каскадного расположения окон нескольких редакторов.
- **Упорядочить горизонтально (Tile Horizontally)**: стандартная команда Windows последовательного размещения всех окон по горизонтали.
- **Упорядочить вертикально (Tile Vertically)**: стандартная команда Windows последовательного размещения всех окон по вертикали.
- **Закрыть все (Close All)**: команда закрытия окон всех редакторов. Если в каком-либо из них производились изменения, система предложит сохранить их.
- **Окна (Windows)**: при выполнении данной команды появляется окно, в котором выполняется манипулирование окнами редакторов. Данное меню доступно, если открыто окно хотя бы одного редактора.
- **<Список открытых окон>**: перечень всех окон редакторов объектов. Данный список отображается, если открывалось окно хотя бы одного редактора.


















Меню **Справка (Help)**

- **Разделы справки (Help Topics)**: команда открытия справочного файла ИСР.
- **Справка по объектам (Object Help)**: команда отображения справочной информации о выделенном объекте.
- **Об ИСР ArchestrA (About ArchestrA IDE)**: команда открытия окна **Об ИСР ArchestrA (About ArchestrA IDE)**, в которое выводятся сведения об охране авторских прав, номер версии программного обеспечения и другая информация об ИСР ArchestrA.

## Панель инструментов

На панели инструментов ИСР находится несколько кнопок, запускающих наиболее часто выполняемые команды. Каждой из кнопок соответствует отдельная команда меню. Описание кнопок и запускаемых ими команд приведено в следующей таблице.

Значок	Отображаемая подсказка	Эквивалентная команда меню
	Change Galaxy (Сменить Galaxy)	Команда Change Galaxy меню Galaxy

	Import Automation Object(s) (Импортирование объектов автоматизации)	Команда Automation Object(s) вложенного меню Import (меню Galaxy)
	Open (Открыть)	Команда Open меню Galaxy
	Save (Сохранить)	Команда Save меню Galaxy
	Find (Поиск)	Команда Find меню Edit
	Check Out (Захват)	Команда Check Out меню Object
	Check In (Освободить)	Команда Check In меню Object
	Undo Check Out (Отменить захват)	Команда Undo Check Out меню Object
	Properties (Свойства)	Команда Properties меню Galaxy
	Deploy (Переслать)	Команда Deploy меню Object
	Undeploy (Отменить использование)	Команда Undeploy меню Object
	Delete (Удалить)	Команда Delete меню Edit
	Customize Toolsets (Настройка наборов инструментов)	Команда Customize Toolsets вложенного меню Configure (меню Galaxy)
	User Information (Сведения о пользователе)	Команда User Information меню Edit
	Galaxy Status (Состояние Galaxy)	Команда Galaxy Status меню Galaxy
	Template Toolbox (Панель шаблонов)	
	Application Views (Структура приложения)	
	Help Topics (Разделы справки)	Команда Help Topics меню Help

Набор кнопок на панели инструментов меняется в зависимости от того, какая панель главного окна ИСР выделена в текущий момент, имеет ли пользователь право выполнять соответствующие действия, были ли произведены изменения в значениях конфигурационных параметров и т.д.

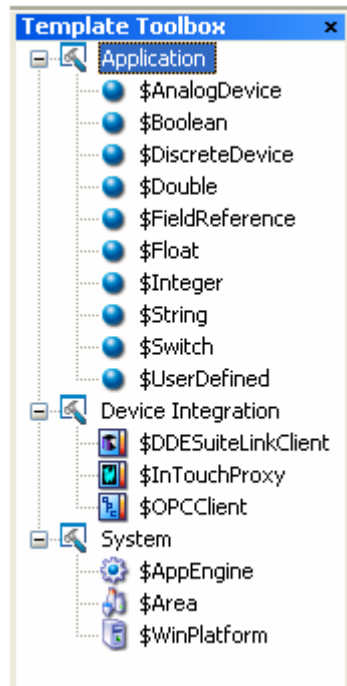
## Панель шаблонов

В этой панели отображаются сведения о шаблонах объектов Galaxy. Перечень шаблонов представлен в панели в виде списка категорий с иерархической структурой. Чтобы раскрыть категорию шаблонов, следует дважды щёлкнуть кнопкой мыши её имя.

По умолчанию, список категорий шаблонов отображается в свернутом состоянии. После регистрации пользователя в системе представление



списка в Панели шаблонов изменяется на представление, которое было во время последнего сеанса.



Созданная Galaxy автоматически заполняется шаблонами, показанными на рисунке.

Именам шаблонов предшествует знак доллара "\$". Каждый шаблон отличается уникальным набором функциональных возможностей и может служить в качестве "строительного" элемента любой прикладной системы. На основе шаблонов создаются экземпляры объектов. Каждый экземпляр объекта получает набор атрибутов, вместе с их значениями по умолчанию, который был определён для соответствующего шаблона. В некоторых случаях значения атрибутов в экземпляре могут быть изменены. Подобная структура "шаблон-экземпляр" является основой надёжного и устойчивого функционирования ИСП.

## Меню "правой" кнопки для Панели шаблонов

Меню, отображаемое при щёлчке правой кнопкой мыши внутри Панели шаблонов, содержит следующие команды (описание команд см. в предыдущем параграфе):

- Open.
- Open Read-Only.
- Check Out.
- Check In.
- Undo Check Out.
- Override Check Out.
- Validate.
- New.
  - Instance.
  - Derived Template.
- Delete.

- Rename.
- Assign To.
- Unassign.
- Export.
  - Automation Object(s).
  - Galaxy Dump.
- Object Help.
- Properties.

## Структура приложения

В панели **Application Views (Структура приложения)** информация об объектах Galaxy может отображаться тремя способами: в виде **Структуры использования (Deployment View)**, в виде **Общей структуры (Model View)** или в виде **Структуры вывода (Derivation View)**. При первом запуске ИСР на экране отображается общая структура.

---

**Примечание.** В каждой структуре приложения названия стандартных объектов выделяются полужирным шрифтом. Стандартные объекты представляет собой зоны (Area), WinPlatform и объекты AppEngines, с которыми связываются новые экземпляры объектов (в структуре использования Deployment View или в общей структуре Model View). Конкретный объект определяется типом создаваемого экземпляра. Например, при создании экземпляра объекта ApplicationObject он автоматически связывается с зоной (Area) по умолчанию, а создаваемый экземпляр объекта AppEngine будет связан с WinPlatform и зоной по умолчанию. Созданный экземпляр можно связать с другим объектом, перетащив его с помощью мыши на соответствующий элемент.

---

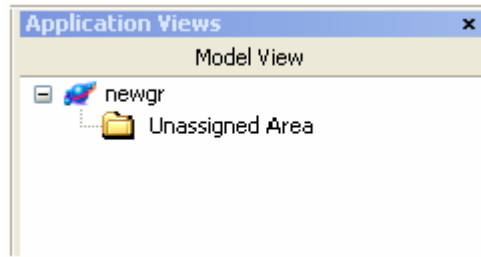
Для каждого объекта в каждой структуре отображается следующая информация:

- Вид использования объекта: not deployed (не используется), deployed (используется), pending configuration update (обновление конфигурации) и pending software update (обновление программного обеспечения). "Обновление конфигурации" означает, что объект в текущий момент используется, и что его параметры с момента рассылки в места использования были изменены. "Обновление программного обеспечения" означает, что объект используется, и что программные модули в составе его шаблона были изменены.
- Состояние конфигурации объекта: Good (нормальное), warning (предупреждение), Bad (ошибка).
- Состояние захвата объекта: checked out (захвачен), checked in (освобождён).

При переходе из одной структуры приложения в другую объект, выделенный в одной из них, автоматически будет выделен и в другой, если только он в ней существует. Например, шаблоны не отображаются в структуре использования и в общей структуре.

## Общая структура

В общей структуре (Model View) все объекты представлены с учётом их взаимосвязи (физического местоположения или связи типа вложение) в виде каталоговой структуры. Эта структура наиболее точно отражает прикладную структуру производственных процессов, например взаимосвязь технологических зон, накопителей, вентилялей, насосов и т.д.



Отображаемый список по функциональным возможностям аналогичен дереву Проводника Windows и первоначально содержит следующие элементы: имена Galaxy и папки **Unassigned Area (Нераспределённые объекты)**.

В общей структуре все объекты сгруппированы по зонам и вложенности. Взаимосвязи объектов отображаются следующим образом:

- Вершиной дерева является Galaxy.
- На следующем уровне отображаются основные зоны.
- В каждой основной зоне отображаются все входящие в неё зоны. Допускается многоуровневое вложение.
- В каждой зоне отображаются все входящие в неё объекты.
- Вложенные объекты отображаются в соответствующем контейнере. Допускается многоуровневое вложение,

---

**Примечание.** Вложенные объекты принадлежат той же зоне, что и содержащие их объекты. Кроме того, некоторые имена в развёрнутом древовидном списке могут быть усечены. Чтобы увидеть их полностью, выделите объект и выполните команду **Properties (Свойства)** меню **Galaxy**. Появится окно свойств объекта, в котором его имя будет представлено в неусечённом виде.

---

- Объекты, которые в текущий момент не включены в состав ни одной из зон, перечислены в папке **Нераспределённые объекты (Unassigned Area)**. При этом все взаимосвязи между объектами – "родительский-дочерний" – сохраняются.

Список объектов на каждом уровне древовидного списка выводится в алфавитном порядке. Названия стандартных объектов выделяются полужирным шрифтом.

Чтобы связать один объект с другим, выделите его и перетащите на целевой объект. Чтобы разорвать связь объектов, перетащите нужный объект на другой объект или в папку нераспределённых объектов. Связь несоответствующих по типу объектов не допускается (на экране отображается символ недопустимости этой операции).

---

**Примечание.** Допускается перетаскивание экземпляров объектов из любой структуры приложения в область результатов поиска окна **Найти (Find)**. Это позволяет уточнять критерии поиска объектов, с которыми возможна организация связи.

---

Подробнее о вложении объектов см. главу "Вложение объектов".

---

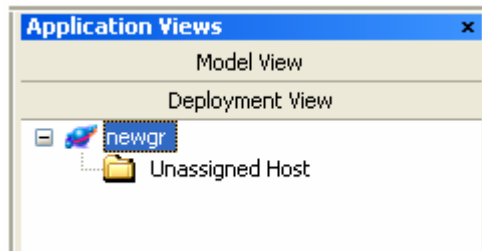
**Примечание.** При установлении связи объекта с другой зоной нужно предварительно отменить его использование в предыдущей зоне.

---

## Структура использования

В структуре использования (Deployment View) все объекты представлены с точки зрения их связей в виде каталоговой структуры. Она показывает, какие объекты находятся на каких компьютерах и в каких объектах

содержатся. Так как в ArchestrA физическое расположение объектов не обязательно должно отражать реальную обстановку, данная структура не предназначена для представления производственной системы.



Отображаемый список по функциональным возможностям аналогичен дереву Проводника Windows и первоначально содержит следующие элементы: имена Galaxy и папки **Нераспределённые хосты (Unassigned Host)**.

В структуре использования все объекты сгруппированы по пунктам назначения следующим образом:

- Вершиной дерева является Galaxy.
- На следующем уровне отображаются все WinPlatform.
- В папке каждой платформы отображаются все связанные с ней объекты AppEngine.
- В папке каждого объекта AppEngine отображаются связанные с ним зоны и DI-объекты (например объекты DINetwork).
- В папке каждой зоны отображаются все связанные с ней объекты, такие как ApplicationObject.
- В папке каждого объекта ApplicationObject отображаются все вложенные объекты ApplicationObject. Допускается многоуровневая вложенность.
- В папке каждого объекта DINetwork отображаются связанные с ним объекты DIDevice.
- Не связанные объекты перечислены в папке нераспределённых хостов. Группирование объектов в ней выполнено по зонам и вложенности.

---

**Внимание!** Объекты DINetwork характеризуются особыми конфигурационными ограничениями (например, может ли на одной платформе WinPlatform использоваться более одного объекта или нет). ИСП ArchestrA не проверяет выполнение этих ограничений. Описание указанных конфигурационных ограничений см. в справочных файлах для каждого объекта DINetwork.

---

Список объектов на каждом уровне древовидного списка выводится в алфавитном порядке. Имена стандартных объектов выделяются полужирным шрифтом.

Чтобы связать один объект с другим, выделите его и перетащите на целевой объект. Чтобы разорвать связь объектов, перетащите нужный объект на другой хост. Связь не соответствующих по типу объектов не допускается (на экране отображается символ недопустимости операции). Чтобы освободить объект, перетащите его обозначение в папку нераспределённых хостов.

---

**Примечание.** При изменении связи используемого объекта нужно предварительно отменить его использование в предыдущем объекте.

---

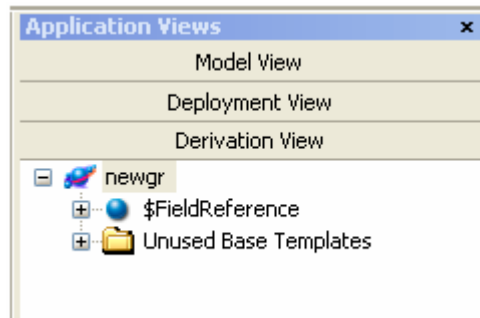
**Примечание.** Допускается перетаскивание экземпляров объектов из любой структуры приложения в область результатов поиска окна **Найти (Find)**.

---

Это позволяет уточнять критерии поиска объектов, с которыми возможна организация связи.

## Структура вывода

В структуре вывода (Derivation View) все объекты и шаблоны представлены с точки зрения их происхождения. То есть в ней показано, из каких объектов получены те или иные объекты системы.



Отображаемый список по функциональным возможностям аналогичен дереву Проводника Windows и первоначально разделён на следующие уровни и подуровни: имя Galaxy, используемые базовые шаблоны (на рисунке показаны как элемент **\$FieldReference**) и папка **Не использованные базовые шаблоны (Unused Base Templates)**. Шаблон **\$FieldReference** отображается по той причине, что в Galaxy уже имеются шаблоны, созданные на его основе.

В структуре вывода все объекты отображаются с учётом их отношений "родительский-дочерний" следующим образом:

- "Вершиной" дерева является Galaxy.
- На следующем уровне отображаются базовые шаблоны вместе с соответствующими производными объектами (шаблонами или экземплярами объектов).
- В папке каждого базового шаблона отображаются все шаблоны и объекты, созданные на его основе. Допускается многоуровневая вложенность. Экземпляры, созданные на основе производных шаблонов, отображаются под соответствующими "родителями".
- Не использованные шаблоны, для которых нет ни производных шаблонов, ни экземпляров объектов, сгруппированы в папке **Не использованные базовые шаблоны (Unused Base Templates)**.

Элементы, названия которых начинаются с символа доллара "\$", представляют собой шаблоны, как базовые, так и производные. Элементы каждого уровня упорядочены по алфавиту. Имена стандартных объектов отображаются полужирным шрифтом.

Как и в других структурах, перетаскивание объекта дерева в другое место приводит к созданию между ними связи соответствующего типа.

**Примечание.** Допускается перетаскивание экземпляров объектов из любой структуры приложения в область результатов поиска окна **Найти (Find)**. Это позволяет уточнять критерии поиска объектов, с которыми возможна организация связи.

## Команды меню правой кнопки

Состав меню правой кнопки для каждой из структур определяется её типом. Некоторые из перечисленных команд в панели той или иной структуры будут недоступны. Подробнее о командах см. параграф "Панель меню".

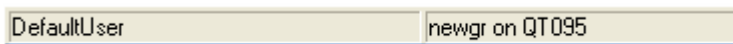
- Open.
- Open Read-Only.
- Check Out.
- Check In.
- Undo Check Out.
- Override Check Out.
- Validate.
- New.
  - Instance.
  - Derived Template.
- Delete.
- Rename.
- Rename Contained Name.
- Deploy.
- Undeploy.
- Upload Runtime Changes.
- Assign To.
- Unassign.
- Set as Default.
- Export.
  - Automation Object(s) ;
  - Galaxy Dump;
- Object Help.
- View in Object Viewer.
- Properties.

## Редактор объектов

Данная часть главного окна ИСП предназначена для отображения окон редакторов конфигурации объектов, показанных в панелях шаблонов и структуры приложения. Состав каждого из этих окон (кнопки, данные и команды меню) определяется типом выбранного объекта. Кроме того, в окно редактора может быть выведена справочная информация о назначении и конфигурационных параметрах выбранного объекта.

## Панель состояния

Сведения, отображаемые в панели состояния, определяются текущим местоположением курсора мыши. Если он находится поверх команды меню, в панели состояния выводится её краткое описание. В панели состояния также отображается имя пользователя, имя Galaxy, в которой он зарегистрировался, и название Репозитория Galaxy.



## Панель операций

В панели операций отображается ход и результаты выполнения операций с базой данных Galaxy, которые при этом могут выполняться одновременно с другими действиями по разработке приложения. В настоящее время проверка конфигурации объектов является единственной операцией, результаты которой выводятся в данную панель.

---

**Внимание!** Допускается проверка конфигурации как шаблонов, так и экземпляров объектов, но только тех, которые не захвачены.

---

Проверка конфигурации включает в себя такие действия, как проверка допустимости указанных значений атрибутов, компиляция скриптов, обновление и разрешение ссылок, проверка расширений, обновление состояния, а также проверка прочих конфигурационных параметров, определяемых типом данного объекта.

---

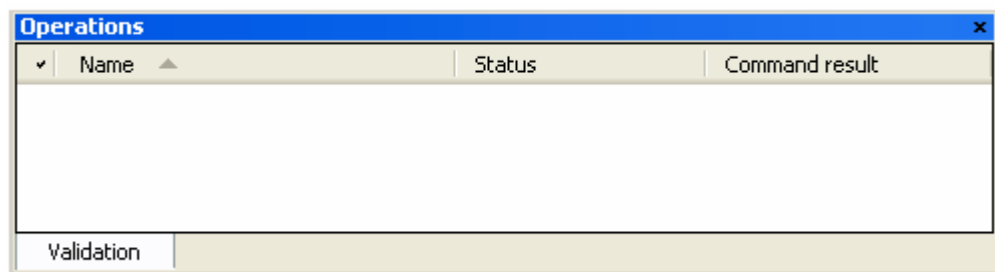
**Примечание.** Основное назначение указанного действия заключается в проверке правильности конфигурации объектов до импортирования соответствующих библиотек скриптов. Некоторые из объектов системы могут находиться в состоянии "Плохое" ("Bad"). В результате проверки ошибочные ссылки на функции библиотеки скриптов исправляются, а состояние объекта изменяется на "Годное" ("Good").

---

Чтобы панель операций появилась на экране, выполните одно из следующих действий:

- Щёлкните объект правой кнопкой мыши (допускается выделение нескольких объектов) и выполните в открывшемся меню команду **Проверить (Validate)**.
- Выполните команду **Операции (Operations)** меню **Вид (View)**.

В главном окне появится следующая панель:



Чтобы скрыть эту панель, щёлкните символ "X" в её правом верхнем углу.

В процессе проверки в панели отобразится значок и имя проверяемого объекта, а также ход выполнения операции. Состояние объекта представляется значком в столбце **Статус (Status)**: его отсутствие означает удовлетворительное состояние, значок ошибки или предупреждения – соответствующее неудовлетворительное состояние. Результат проверки выводится в столбце **Command Result (Результат команды)**: "Завершено" ("Succeeded") или "Ошибка" ("Failed"), причём этом в данный столбец могут быть выведены дополнительные сведения о проверке.

---

**Примечание.** Выполнение команды Проверить (Validate) для объекта Galaxy приводит к проверке конфигурации всех объектов системы. В этом случае столбец Command Result (Command Result) заполнится значениями только в самом конце операции.

---

При проверке нескольких объектов их список будет отсортирован по именам объектов. Щёлчок кнопкой мыши заголовка столбца изменяет порядок сортировки: по именам или по значкам. Щелчок кнопкой мыши заголовка столбца с "галочкой" позволяет сортировать объекты, которые

являются захваченными и, таким образом, не входят в круг проверки. Если объект захвачен, его значок будет отображаться вместе с "галочкой".

После выделения объекта в панели операций с ним можно выполнять такие же действия, как и в панели шаблонов или панели структуры приложения. Для этого нужно щёлкнуть его правой кнопкой мыши и выбрать в появившемся меню нужную команду. После двойного щелчка кнопкой мыши имени объекта появляется окно соответствующего редактора конфигурации. Двойной щелчок значка состояния приводит к открытию страницы **Ошибки и предупреждения (Errors/Warnings)** окна свойств объекта. Отображаемый в панели операций текст можно переписать в Буфер обмена операционной системы, выполнив команду **Копировать (Сору)** меню правой кнопки или нажав сочетание клавиш **CTRL+C**. При изменении состояния объектов Galaxy содержимое панели операций также обновляется.

## Настройка рабочей области

При первом запуске ИСР в её главном окне, в верхней части, отображаются панель инструментов, панели шаблонов и структуры приложения (слева рядом друг с другом) и панель редактора объектов (справа). Панель операций вначале не отображается. В последующих сеансах работы этого же пользователя размещение элементов главного окна определяется их положением в момент завершения предыдущего сеанса.

Пользователь может менять положение элементов главного окна ИС любым способом. Панели шаблонов, структуры приложения и операций можно перемещать по поверхности главного окна, располагать одно поверх другого, пристыковывать к любой границе окна и т.д. При двойном щелчке кнопкой мыши заголовка панели её вертикальная ориентация восстанавливается. Скрыть любой из этих элементов можно, нажав кнопку со значком панели шаблонов или панели структуры приложения в панели инструментов или кнопку с символом "X" в заголовке панели операций.

Если главное окно ИСР не максимизировано, панели структуры приложения, шаблонов и операций можно размещать вне его границ на экране.

Окно редактора объектов занимает всё свободное пространство главного окна ИСР, не занятое панелями структуры приложения, шаблонов и операций.

## Комбинации клавиш

В следующей таблице перечислены сочетания клавиш, нажатие которых приводит к выполнению определённых команд ИСР.

Сочетание клавиш	Эквивалентная команда меню
CTRL+N	Экземпляр (Instance) вложенного меню команды Создать (New) меню Galaxy
CTRL+SHIFT+N	Derived Template (Производный шаблон) вложенного меню команды Создать (New) меню Galaxy
CTRL+O	Открыть (Open) меню Galaxy
CTRL+F4	Закреть (Close) меню Galaxy
CTRL+S	Сохранить (Save) меню Galaxy
ALT+ENTER	Свойства (Properties) меню Galaxy
F2	Переименовать (Rename) меню Редактировать (Edit)
SHIFT+F2	Изменить имя вложенного объекта (Rename Contained)



	Name) меню Редактировать (Edit)
DEL	Удалить (Delete) меню Редактировать (Edit)
CTRL+F	Найти (Find) меню Редактировать (Edit)
F1	Разделы справки (Help Topics) меню Справка (Help)
CTRL+F1	Справка по объектам (Object Help) меню Справка (Help)
ALT+F4	Выход (Exit) меню Galaxy
CTRL+Z	Отменить (Undo) меню правой кнопки
CTRL+X	Вырезать (Cut) меню правой кнопки
CTRL+C	Копировать (Copy) меню правой кнопки
CTRL+V или SHIFT+INSERT	Вставить (Paste) меню правой кнопки
SHIFT+DELETE	Удалить (Delete) меню правой кнопки
CTRL+A	Выделить всё (Select All) меню правой кнопки

## Допустимые имена и символы

В следующей таблице приведены правила именования различных элементов ArchestrA. Подробнее об именовании ссылок см. главу "Ссылки". В число указанных в таблице букв входят буквы любых национальных языков. Определение терминов см. в конце таблицы.

Элемент	Максимальная длина названия	Допустимые символы	Первый символ названия	Прочие правила
Имя Galaxy	32	Буквы, цифры и специальные символы	Только буква	-
Название шаблона	32 (включая символ "\$")	Буквы, цифры и специальные символы	\$	Название должно содержать хотя бы одну букву, знак доллара не должен быть вторым символом
Название экземпляра	32	Буквы, цифры и специальные символы	Не "\$"	Название должно содержать хотя бы одну букву
Иерархическое имя	329	Буквы, цифры, специальные символы и символ точки "."	Не "\$"	Название должно содержать хотя бы одну букву
Название вложенного элемента	32	Буквы, цифры и специальные символы	Не "\$"	Название должно содержать хотя бы одну букву
Название статического атрибута или атрибута UDA	329 (включая точки)	Буквы, цифры, специальные символы и символ точки "."	-	Название должно содержать хотя бы одну букву

Название динамического атрибута	329	Любой символ за исключением пробела	-	-
Название свойства	32	Буквы, цифры и специальные символы	-	-
Название набора инструментов	64	Любые печатные символы	Не "\$"	-
Имя пользователя	255	Любые печатные символы за исключением "User Name" и специальных символов	-	-
Название группы пользователей	32	Буквы, цифры и специальные символы	Не "\$"	Название должно содержать хотя бы одну букву
Название роли	512	Буквы, цифры и специальные символы	-	-
Имя скрипта	32	Буквы, и/или цифры, и/или специальные символы, и/или "."	-	Название должно содержать хотя бы одну букву
Название поставщика	-	Любые печатные символы за исключением символов ? " / \ < > *   :	-	-
<b>Определение терминов:</b>				
<ul style="list-style-type: none"> <li>• буква = любая буква алфавита любого национального языка</li> <li>• цифра = любой числовой символ</li> <li>• специальный символ: любой графический символ за исключением следующих: символов с ASCII-кодами от 0 до 32 (неграфические символы); символов . + - * / \ = ( ) ` ~ ! % ^ &amp; @ [ ] { }   : ; ' , &lt; &gt; ? " &lt;пробел&gt;</li> <li>• специальные символы в имени пользователя: = / \ [ ] :   &lt; &gt; + = ; , ? * @ &lt;пробел&gt;</li> </ul>				

**Примечание.** Следующие имена тэгов являются зарезервированными и не могут использоваться в качестве имени пользователя, имени группы пользователей, имени роли, имени Galaxy и названия вложенного объекта: Me, MyContainer, MyArea, MyHost, MyPlatform, MyEngine, System.

## Связь узел-узел

Взаимодействовать друг с другом могут все компьютеры, на которых установлено программное обеспечение с поддержкой Archestra. При организации обмена данными между узлами нужно обеспечить правильное

определение учётных сведений о пользователях ArchestrA и правильное конфигурирование компьютеров с несколькими сетевыми адаптерами.

## Учётные сведения о пользователях ArchestrA

Обмен данными между компьютерами осуществляется с использованием учётных сведений о пользователях ArchestrA, указываемых во время установки каждого компонента ArchestrA на каждом компьютере сети, включая компьютер ИСР.

---

**Внимание!** Учётная запись, разрешающая обмен данными ArchestrA, представляет собой обычную учётную запись пользователя операционной системы Windows на локальном компьютере или домене. Не удаляйте её с помощью системных средств управления учётными записями, так как при этом функционирование ИСР будет невозможно.

---

Если учётные сведения ArchestrA в узле ИСР по какой-либо причине будут удалены, их нужно восстановить с помощью утилиты изменения параметров регистрации в сети ArchestrA (ArchestrA Change Network Account).

---

**Примечание.** Чтобы выполнять какие-либо действия с помощью этой утилиты, нужно обладать административными правами в системе компьютера.

---

### Чтобы восстановить учётную запись пользователя ArchestrA

1. Запустите утилиту изменения параметров регистрации (Change Network Account), нажав кнопку **Start (Пуск)** Панели задач Windows, а затем последовательно поместив курсор мыши на пункты **Programs (Программы)**, **Wonderware** и **Common (Общие)** появляющихся меню и щёлкнув пункт **Изменение сетевых учётных записей (Change Network Account)**.

---

**Примечание.** Подробнее об отображаемых в окне утилиты элементах можно узнать, нажав кнопку **Справка (Help)**.

---

2. В системе ArchestrA, состоящей из одного узла, можно определить любую учётную запись.
3. В системе с несколькими узлами в восстанавливаемых учётных данных нужно указать такое же имя пользователя и пароль, что и во всех других компьютерах с установленным программным обеспечением ArchestrA.
4. Щёлкните **ОК**.

---

**Примечание.** После ввода учётных данных в окне утилиты, прежде чем информация на локальном компьютере будет обновлена, может пройти несколько минут. В течение этого периода функционирование ИСР может быть нарушено. При повторной загрузке узла Galaxy эти изменения вступают в силу немедленно.

---

## Компьютеры с несколькими сетевыми адаптерами

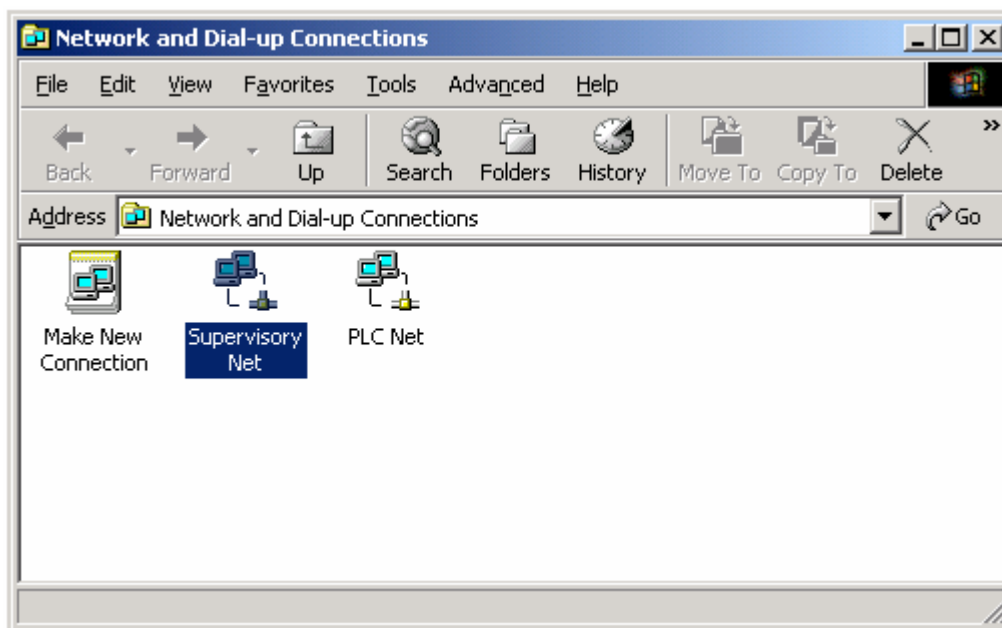
Для обеспечения обмена данными с другими узлами ArchestrA любой узел Galaxy, имеющий более одного сетевого адаптера, должен быть сконфигурирован в соответствии с инструкциями настоящего параграфа. Если сетевые адаптеры установлены с целью резервирования, дополнительно прочитайте параграф "Резервирование в платформах WinPlatform".

---

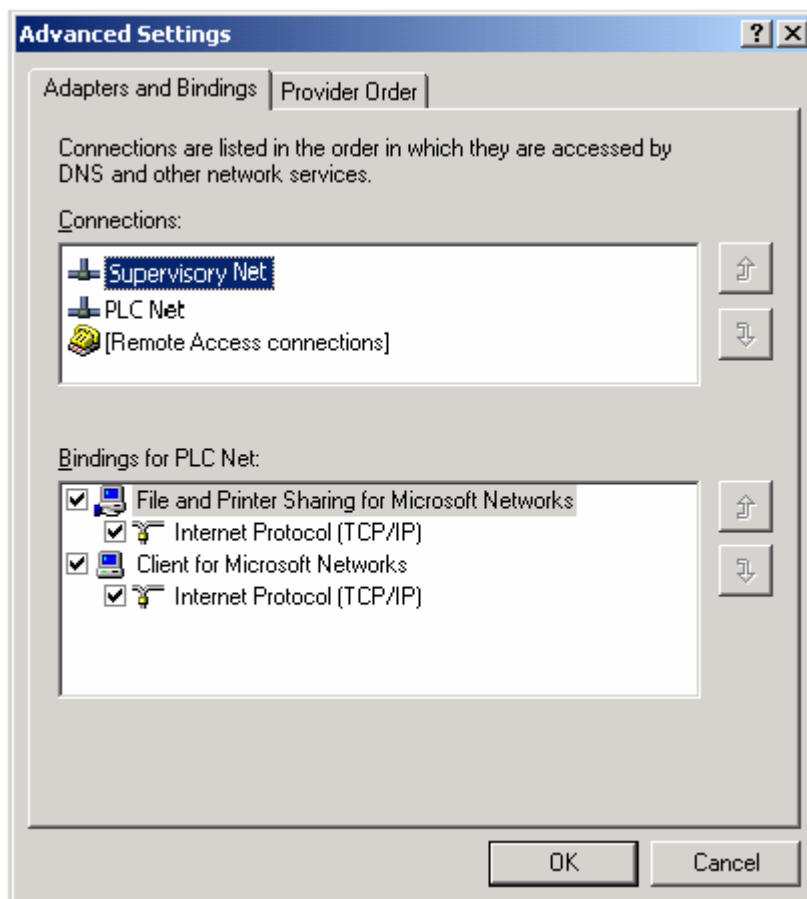
**Внимание!** Если в Galaxy используется только один из нескольких установленных на компьютере сетевых адаптеров, нужно отключить их все, за исключением входящих в состав управляющей сети.

---

На компьютере с несколькими сетевыми адаптерами прежде всего нужно определить правильный порядок сетевых соединений. Для этого откройте (см. следующий рисунок) окно **Сетевые подключения (Network and Dial-Up Connections)** и измените наименование каждого адаптера в соответствии с его назначением (например "Supervisory Net" или "PLC Net"). В разных операционных системах это окно может открываться по-разному. Например, в **Windows 2000 Professional** для этого нужно нажать кнопку **Пуск (Start)** Панели задач, затем последовательно помещать курсор мыши поверх пунктов **Программы (Programs)**, **Стандартные (Accessories)** и **Связь (Communications)** появляющихся меню и щёлкнуть пункт **Network and Dial-Up Connections**.



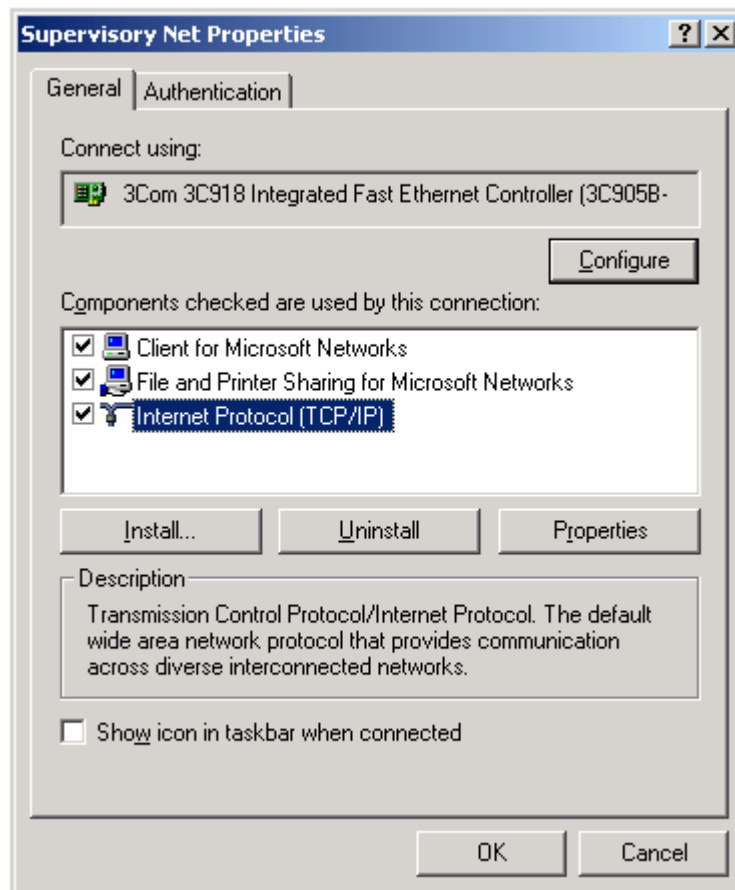
Чтобы упорядочить сетевые адаптеры должным образом, выполните команду **Дополнительные установки (Advanced Settings)** меню **Дополнительно (Advanced)** окна сетевых подключений. Измените в этом окне порядок указания сетей в панели **Соединения (Connections)** с помощью кнопок с обозначениями стрелок вверх и вниз и щёлкните **ОК**. Первым в списке должно быть соединение с управляющей сетью. Если на компьютере установлено более двух сетевых адаптеров (например для подключения к управляющей сети, к сети ПЛК и для резервирования соединения ArchestrA), остальные могут указываться в любом порядке.



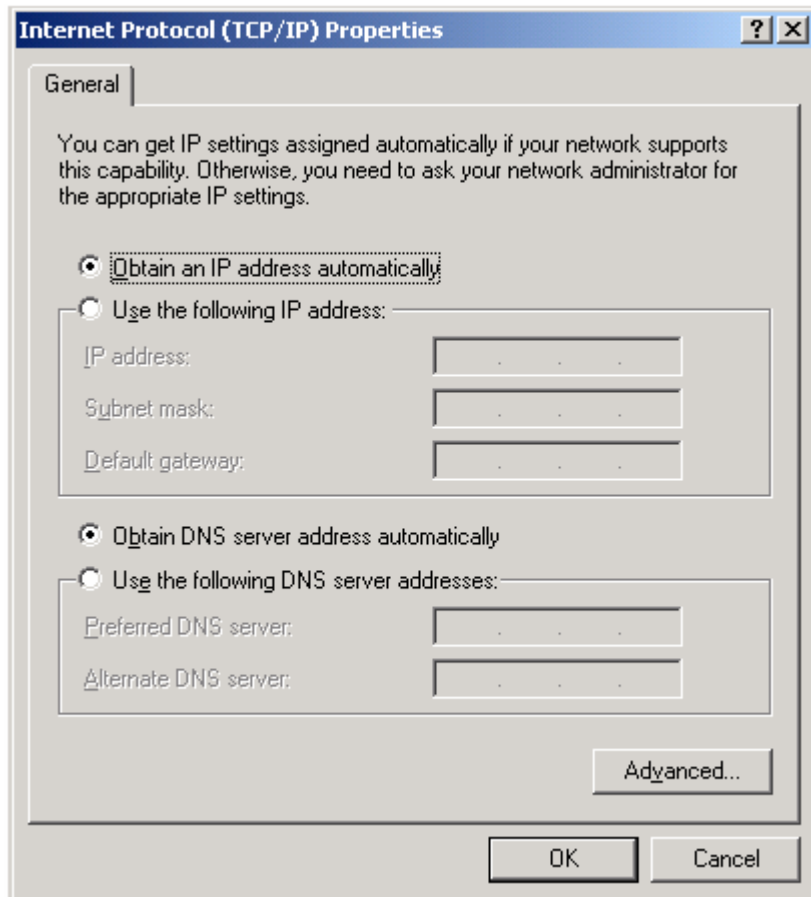
Для успешного обмена данными между узлами ArchestrA необходима установка значений некоторых других параметров: IP-адреса и DNS-параметров каждого сетевого адаптера.

**Выполните следующие действия для конфигурирования каждого сетевого адаптера:**

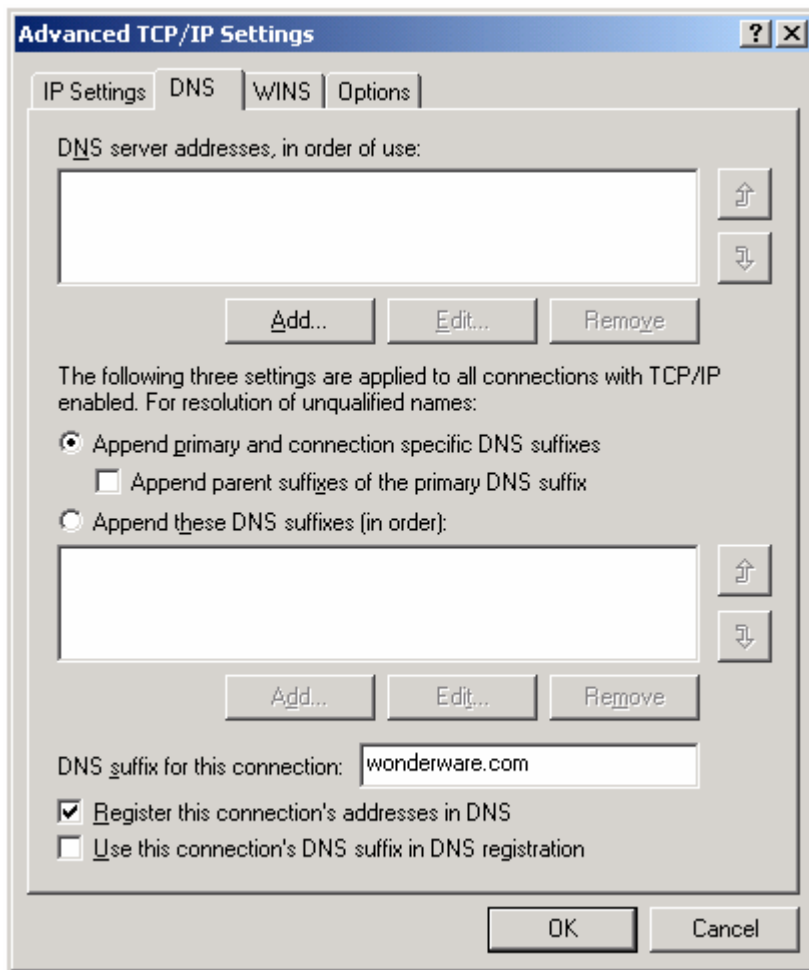
1. Щёлкните правой кнопкой мыши обозначение сетевого соединения в окне сетевых подключений и в появившемся меню выполните команду **Свойства (Properties)**. Появится окно свойств выбранного сетевого соединения:



2. Отметьте в списке компонентов, используемых для поддержки указанного сетевого соединения, пункт **Протокол Интернет, TCP/IP (Internet Protocol, TCP/IP)** и щёлкните **Свойства (Properties)**. Появится окно свойств протокола Интернет:



3. Для соединения с управляющей сетью установите флажок **Получать IP-адрес автоматически (Obtain an IP address automatically)**. Для сетей других типов установите флажок **Использовать следующий IP-адрес (Use the following IP address)**. Требуемые сведения для установки полей этой группы получите у своего системного администратора.
4. Щёлкните **Дополнительно (Advanced)**. Появится окно **Дополнительные параметры TCP/IP (Advanced TCP/IP Settings)**. Перейдите на страницу с закладкой DNS.



5. Для соединения с управляющей сетью установите флажок **Регистрировать адреса этого соединения в службе DNS (Register this connection's addresses in DNS)**, в остальных случаях – сбросьте его.
6. Щёлкните **ОК**.

## Требования к минимальному дисковому пространству

После установки программного обеспечения сервера промышленных приложений IAS для выполнения некоторых операций на диске должно быть не менее 100 Мбайт свободного пространства. В число этих операций входят создание Galaxy, установка объектов, импортирование и экспортирование объектов, загрузка и выгрузка базы данных Galaxy, а также резервное копирование и восстановление Galaxy.

Указанное требование к свободному дисковому пространству применимо как к узлу Galaxy, так и любому другому удалённому узлу ИСР.

Если на диске компьютера будет менее 100 Мбайт свободного пространства, выполнение этих действий может завершаться с ошибкой.



## ГЛАВА 2

# Объекты

Объекты в среде ArchestrA разделены на две категории: шаблоны и экземпляры. Шаблоны представляют собой типы элементов, из которых может быть создана пользовательская Galaxy. Экземпляры объектов, создаваемые по шаблонам, представляют собой уникальные компоненты рабочей системы. Прежде чем использовать их на соответствующих узлах системы, эти объекты должны быть должным образом сконфигурированы и связаны друг с другом.

В настоящей главе описаны методы создания, импортирования, конфигурирования шаблонов и экземпляров объектов, а также методы манипулирования наборами шаблонов в Панели шаблонов ИСР.

## Содержание

- Объекты
- Импортирование объектов
- Экспортирование объектов
- Конфигурирование объектов
- Шаблоны объектов
- Управление шаблонами
- Создание шаблонов на основе других шаблонов
- Экземпляры объектов
- Импортирование файлов определений объектов
- Экспортирование определений объектов в файлы
- Создание экземпляров объектов по шаблонам
- Проверка допустимости конфигурации объектов
- Построение приложения
- Пересылка объектов в узлы использования
- Повторная пересылка объектов
- Отмена использования объектов

## Объекты

В ИСР могут использоваться любые объекты, созданные с помощью инструментального пакета разработки объектов ArchestrA Object Toolkit. Термин "объекты" означает шаблоны, производные шаблоны и экземпляры объектов, создаваемые на основе этих шаблонов.

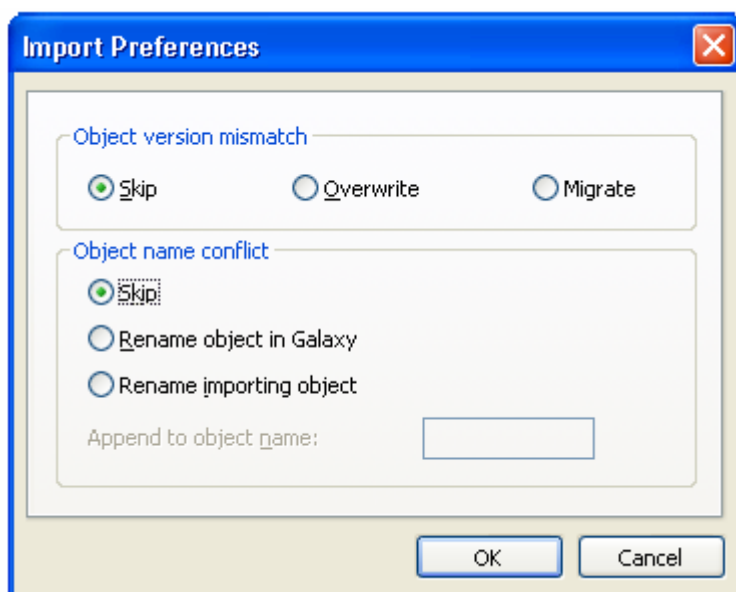
## Импортирование объектов

Чтобы использовать объект, его нужно предварительно импортировать в Galaxy с помощью функции импортирования объектов.

**Примечание.** Можно также импортировать экземпляры объектов, ранее экспортированные из системы. При этом их прежние определения, такие как привязки, вложенности, зоны и т.д., по возможности сохраняются.

Объекты импортируются из специальных пакетных файлов ArchestrA, как правило, с расширениями .aaPKG или .aaPDF. В одном пакетном файле может содержаться несколько объектов.

Запись в базу данных Galaxy определений нескольких объектов с совпадающими именами или нескольких копий с одной и той же версией объекта невозможна. Способы разрешения конфликтов имён и версий объектов при импортировании объектов указываются в окне **Параметры импортирования (Import Preferences)**.



Способ разрешения конфликта версий (импортировать ли обновлённую версию объекта, определение которого уже есть в базе данных) указывается в панели **Несоответствие версии объекта (Object Version Mismatch)**. Обновление версии может быть результатом изменения значения параметра MinorVersion или ConfigVersion. Выбор пункта **Пропустить (Skip)** означает отмену импортирования нового объекта с сохранением в базе данных определения существующего объекта, выбор пункта **Переписать (Overwrite)** – замену определения существующего объекта его обновлённой версией и выбор пункта **Миграция (Migrate)** – импортирование выбранных объектов, обновлённых коренным образом. Последнее обычно случается после установки новой версии программного обеспечения IAS (если Galaxy не обновлена и выбран пункт **Мигрировать – Migrate**, импортирование объектов будет выполняться так же, как и при выборе пункта **Пропустить – Skip**) или при импортировании дополнительных объектов, обновлённых коренным образом.

**Внимание!** После пропуска объекта с несоответствующей версией процесс импортирования объектов из пакетного файла продолжается.

В панели **Конфликт имён объектов (Object Name Conflict)** указывается способ разрешения ситуации, когда имена импортируемого и существующего в Galaxy объектов совпадают. Пункт **Пропуск (Skip)** означает отмену импортирования определения объекта с совпадающим именем. При выборе пункта **Переименовать объект в Galaxy (Rename**

**Object in Galaxy**) изменяется имя существующего в Galaxy объекта: к нему добавляется строка (длиной не более четырёх символов, по умолчанию "\_old"), указанная в поле **Добавить к названию объекта (Append to Object Name)**. При выборе пункта **Переименовать импортируемый объект (Rename Importing Object)** изменяется имя импортируемого объекта: к нему добавляется строка (длиной не более четырёх символов, по умолчанию "\_new"), указанная в поле **Добавить к названию объекта (Append to Object Name)**.

---

**Примечание.** Процедура разрешения конфликтов именования действительна только для шаблонов и экземпляров объектов, созданных по различающимся базовым шаблонам (то есть имеющих различный программный код).

---

В ходе импортирования отображается окно с информацией о состоянии процесса: был ли объект импортирован успешно, был ли пропущен, переименован или перезаписан.

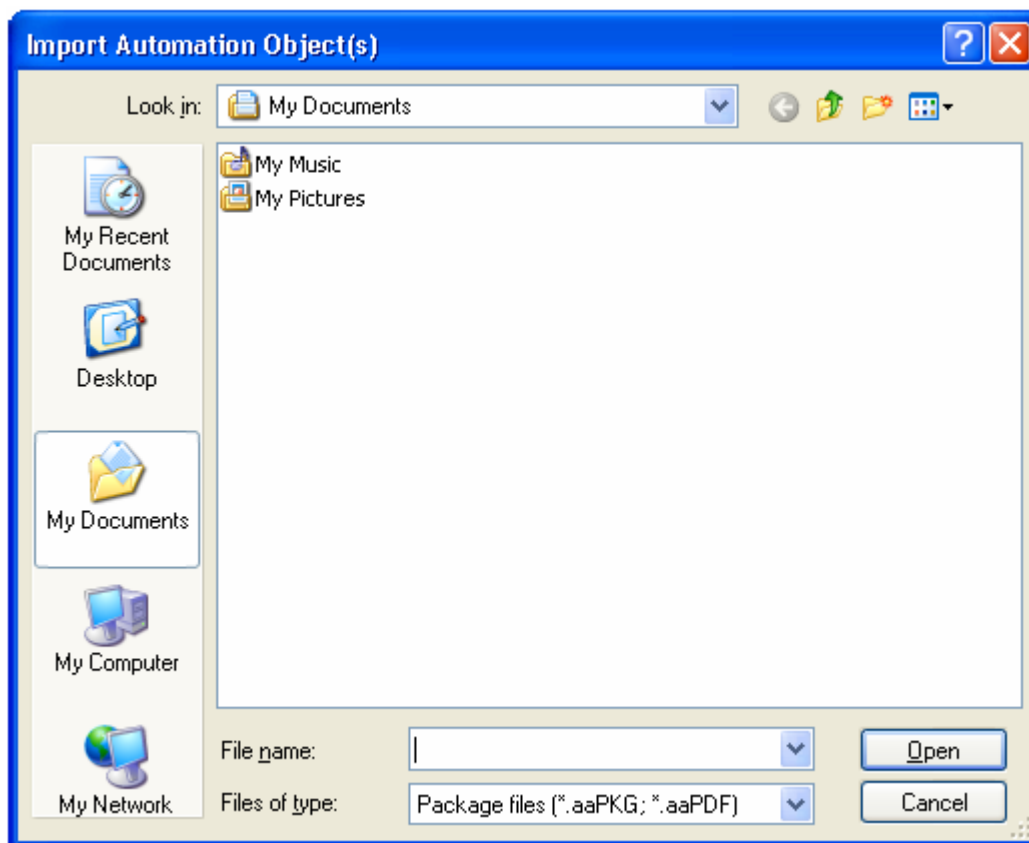
---

**Внимание!** Функциональные возможности объекта могут быть расширены путём присоединения к нему скрипта. В некоторых скриптах могут использоваться функции, код которых хранится во внешнем файле, называемом библиотекой скриптовых функций. Несмотря на то, что во время импортирования эти скрипты также копируются, сами библиотеки нужно импортировать в систему отдельно. Если в скрипте импортируемого объекта будет содержаться ссылка на библиотеку, которой в Galaxy нет, статус объекта будет изменён на "Bad" (плохой), и использовать объект будет невозможно. Исправить данную ситуацию можно, импортировав необходимую библиотеку функций и выполнив процедуру проверки конфигурации объекта. Подробнее о функциях см. параграф "Использование функций в скриптах", о проверке конфигурации – параграф "Проверка допустимости конфигурации объектов".

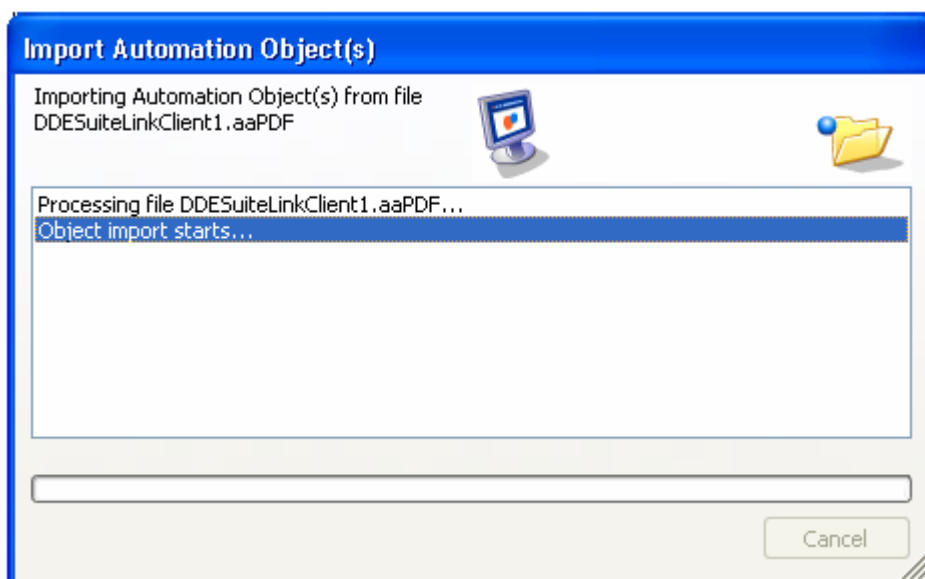
---

#### **Чтобы импортировать объект:**

1. Поместите курсор мыши на пункт **Импортирование (Import)** меню Galaxy и щёлкните пункт **Объекты автоматизации (Automation Objects)** вложенного меню. Появится окно **Импортирование объектов AutomationObject (Import AutomationObjects)**.



2. Найдите нужный пакетный файл с расширением .aaPKG или .aaPDF. Одновременно можно выделить несколько пакетных файлов. Затем щёлкните **Открыть (Open)**. Появится окно **Параметры импортирования (Import Preferences)**.
3. Укажите в окне параметров импортирования способы разрешения конфликтов имён и версий и щёлкните **ОК** для запуска процесса импортирования или кнопку **Отмена (Cancel)** для его отмены. В первом случае появится окно, в котором будет отображаться ход импортирования объектов.



4. После завершения процесса щёлкните **Заккрыть (Close)**.

Импортированные шаблоны будут включены в группы шаблонов, указанные в определении объекта. Импортированные экземпляры

отображаются в списке панели структуры приложения. При этом будут выполнены следующие действия:

- Если ещё набор не существует, он будет создан.
- Если параметры доступа к объекту не совпадают с параметрами ни одной из определённых групп доступа, он будет связан с группой **По умолчанию (Default)**.
- Если объект входит в зону, которая ещё не определена, он будет помещён в папку **Нераспределённые объекты (Unassigned Area)**.
- Если хост-объект, с которым связан импортированный объект, ещё не определён, он будет помещён в папку **Нераспределённые хосты (Unassigned Host)**.

---

**Примечание.** При импортировании новой версии существующего экземпляра он будет помечен как требующий рассылки, если существующий объект используется.

---

## Экспортирование объектов

Определения объектов из одной Galaxy могут быть переданы в другие системы с помощью функции экспортирования. В создаваемом этой утилитой файле (с расширением .aaPKG) будут записаны определения выбранных объектов, соответствующих шаблонов и состояние их конфигураций. Впоследствии эта информация может быть импортирована как в ту же самую, так и другую Galaxy.

При последующих операциях экспортирования в том же сеансе работы с ИСР файл будет создаваться в той же папке, что и ранее. На экране будет появляться окно с запросом на подтверждение перезаписи существующего файла.

Если объект, предназначенный для экспортирования, будет захвачен, в файл будут записаны сведения, соответствующие его "свободному" состоянию.

---

**Внимание!** Функциональные возможности объекта могут быть расширены путём присоединения к нему скрипта. В некоторых скриптах могут использоваться функции, код которых хранится во внешнем файле, называемом библиотекой скриптовых функций. Хотя во время экспортирования эти скрипты также копируются, сами библиотеки нужно экспортировать отдельно.

---

Экспортирование всей базы данных Galaxy аналогично её резервному копированию с помощью Менеджера базы данных Galaxy (Galaxy DataBase Manager), за исключением того, что журналы изменений объектов не копируются. Кроме того, при резервном копировании сохраняется вся структура управления доступом, в то время как при экспортировании извлекаются только те параметры, которые относятся к конкретному объекту.

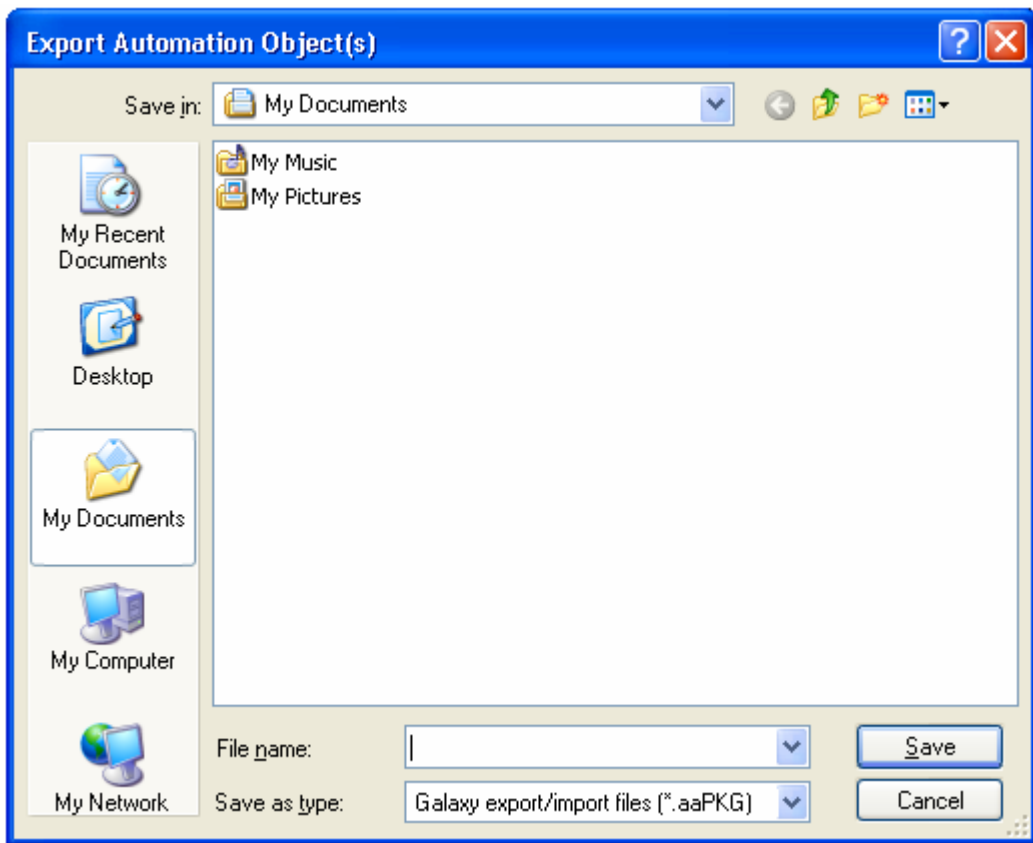
### Чтобы экспортировать объекты:

1. Выделите объект в панели шаблонов или структуры приложения.
2. Поместите курсор мыши на пункт **Экспортирование (Export)** меню **Galaxy** и щёлкните пункт **Объекты автоматизации (Automation Objects)** вложенного меню. Появится окно **Экспортирование объектов AutomationObject (Export AutomationObjects)**.

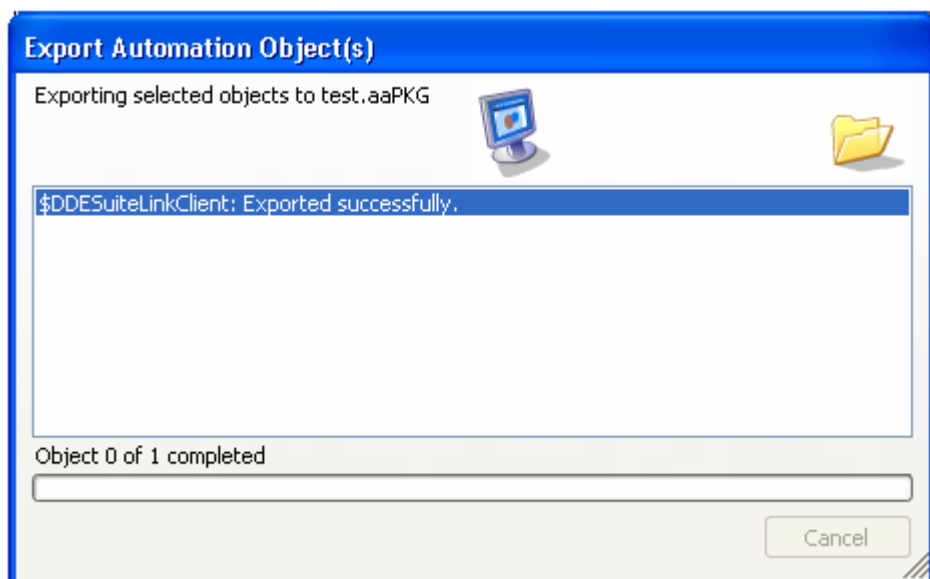
---

**Примечание.** Отметить несколько объектов для экспортирования можно, щёлкая их названия кнопкой мыши при нажатых клавишах **SHIFT** или **CTRL**. Чтобы экспортировать определения всех объектов

Galaxy, выполните команду **Все объекты AutomationObject (All AutomationObjects)** меню **Экспортирование (Export)**.



3. Укажите в окне каталог и имя файла (он получит расширение ..aaPKG) и щёлкните **Сохранить (Save)**. Чтобы отменить процедуру экспортирования, щёлкните **Отмена (Cancel)**. Во время экспортирования данных появится окно, в которое будет выводиться ход выполнения указанного действия:



4. По окончании экспортирования щёлкните **Заккрыть (Close)**. Созданный файл с расширением .aaPKG может быть использован для импортирования соответствующих объектов в другую Galaxy.

**Внимание!** При экспортировании объектов информация об их вложенности в другие объекты сохраняется. Если экспортируемый

объект в данный момент был захвачен, в файл будет записана информация, соответствующая его предшествующему "свободному" состоянию.

---

## Конфигурирование объектов

Шаблоны и экземпляры объектов конфигурируются одинаково, с помощью редакторов конфигурации. Чтобы изменить параметры объекта, выделите его и выполните команду **Открыть (Open)** меню **Galaxy**. В главном окне ИСР откроется окно соответствующего редактора объектов.

Подробнее о редакторах шаблонов см. главу "Редакторы объектов".

Чтобы изменить параметры объекта, его нужно сперва захватить (это делается в момент открытия окна редактора автоматически). По окончании конфигурирования следует освободить объект.

## Захват и освобождение объектов

Блокирование объекта для монопольного изменения его конфигурации выполняется путём его захвата. Чтобы остальные пользователи Galaxy смогли увидеть обновлённую версию объекта, его нужно освободить. При этом все выполненные изменения будут сохранены в базе данных Galaxy. Изменение конфигурации объекта можно отменить, выполнив команду **Отменить захват (Undo Check Out)**.

---

**Примечание.** Текущее состояние объектов отображается во всех связанных с Galaxy ИСР. Возможен также просмотр предыстории изменений любого объекта.

---

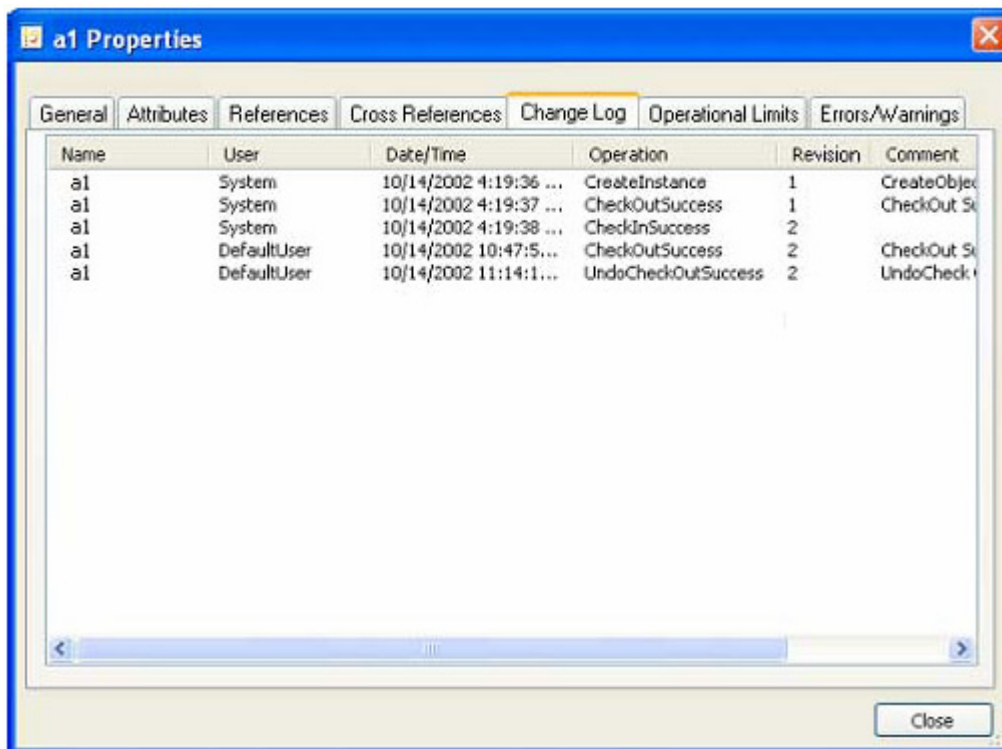
Если объект в текущий момент будет захвачен другим пользователем, команда его захвата будет недоступна.

Одновременно можно выделить несколько объектов для захвата. Если какие-либо из выделенных объектов уже захвачены другим пользователем, появится окно с соответствующей информацией об их состоянии. Если пользователь попытается захватить уже захваченные им объекты, данная команда будет недоступна.

Galaxy помечает захваченные объекты, чтобы их не могли захватить другие пользователи, и записывает соответствующие сведения в журнал изменений объекта. Захваченные объекты выделяются в окне ИСР специальной меткой рядом со значком объекта.

### Чтобы захватить объекты

1. Выделите их в панели шаблонов или структуры приложения.
2. Выполните команду **Захватить (Check Out)** меню **Объект (Object)**. Объект также становится захваченным в момент открытия соответствующего редактора конфигурации. Если объект уже захвачен другим пользователем, редактор запустится в режиме "только чтение". Определить состояние объекта можно, открыв страницу его свойств.

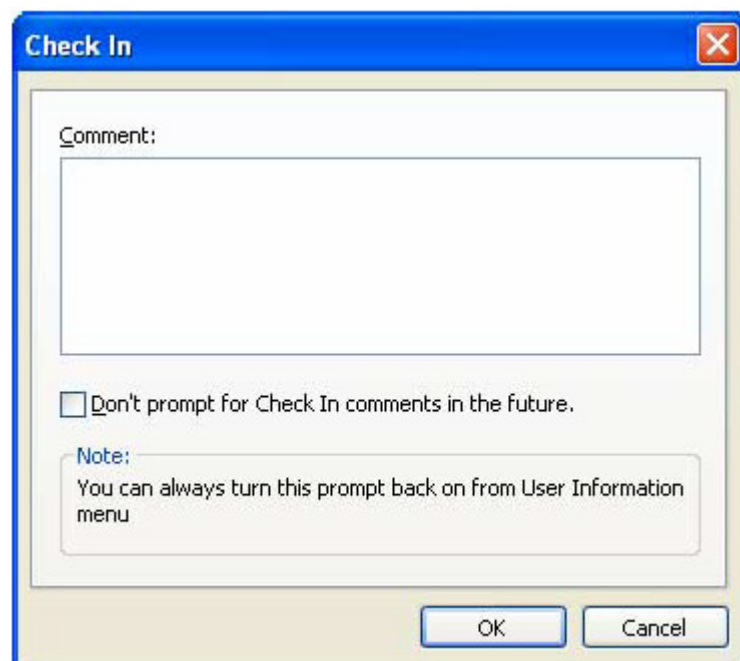


Сведения о пользователе, выполнившего в определённый момент какое-либо действие с объектом, отображаются на странице **Журнал изменений (Change Log)**. Введённые им в окне **Освободить (Check In)** примечания будут показаны в столбце **Комментарий (Comment)**.

#### Чтобы освободить объект

1. Выделите объект в панели шаблонов или структуры приложения и выполните команду **Освободить (Check In)** меню **Объект (Object)**. Появится окно **Освободить (Check In)**.

**Примечание.** Если закрыть окно редактора без выполнения каких-либо изменений в конфигурации объекта, команда освобождения будет аналогична команде отмены захвата. Подробнее см. главу "Редакторы объектов".



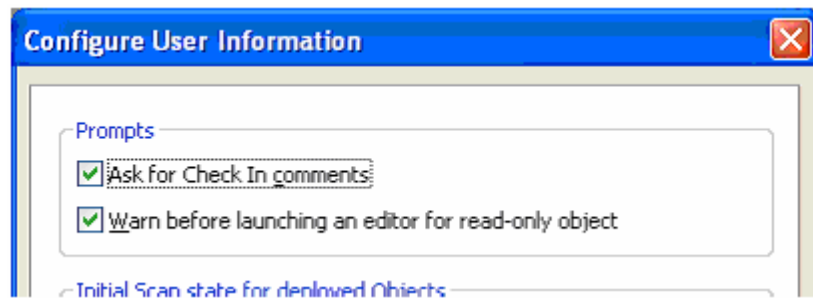


2. Введите, если нужно, примечания и щёлкните **ОК**, чтобы окончательно освободить объект, или кнопку **Отмена (Cancel)**, чтобы приостановить процесс.

Galaxy показывает, являются ли освобождаемые объекты захваченными другими пользователями и запущены ли соответствующие редакторы конфигурации. При попытке освободить уже освобождённый объект команда игнорируется.

В окне **Освободить (Check In)** можно вводить любые примечания и пояснения, относящиеся к изменению конфигурации объекта во время его захвата. Окно содержит следующие элементы:

- **Комментарий (Comment)**: поле ввода примечаний к изменениям в конфигурации объекта;
- **Не открывать окно ввода примечаний в будущем (Don't Prompt for Check-In Comments in the Future)**: после установки этого флажка окно ввода примечаний при освобождении объектов больше не будет открываться. Для отмены запрета нужно будет щёлкнуть пункт **Сведения о пользователе (User Information)** меню **Редактировать (Edit)** и установить флажок **Запрашивать ввод комментариев при освобождении (Ask for Check In Comments)** в появившемся окне **Установка параметров пользователя (Configure User Information)**.



## Команды отмены и игнорирования захвата

В системе имеются две дополнительные команды, управляющие состоянием захвата объекта:

- **Отменить захват (Undo Check Out)**: команда изменения состояния объекта с "захвачен" на "освобождён". После этого любой пользователь может захватить данный объект для изменения его конфигурации. При выполнении данной команды в журнал изменений объекта заносится соответствующая запись. Все изменения, выполненные во время захвата объекта, аннулируются. Если окно редактора конфигурации объектов в этот момент будет открыто, система выведет сообщение об ошибке.
- **Игнорировать захват (Override Check Out)**: команда игнорирования флажка захвата выделенного объекта. Для выполнения данной команды требуется, как правило, наличие особых полномочий (см. главу "Контроль доступа"). Использоваться она должна в моменты, когда конфигурация объекта не изменяется захватившим его пользователем. Если окно редактора конфигурации объектов в этот момент открыто, команда выполнена не будет.

## Удаление объектов

Удалить из Galaxy можно любой шаблон или экземпляр объекта, за исключением следующих случаев. Объект не должен:

- Использоваться.

- Быть родительским шаблоном (по которому созданы экземпляры объектов или производные шаблоны).
- Быть контейнером других объектов.
- Быть захваченным другим пользователем.

#### Чтобы удалить объект из Galaxy

1. Выделите его в панели шаблонов или структуры приложения. Несколько объектов могут быть выделены с помощью мыши и клавиши **Shift** или **Ctrl**.
2. Выполните команду **Удалить (Delete)** меню **Редактировать (Edit)**. Удалены будут те объекты, для которых эта операция допустима.

## Переименование объектов

Объект может иметь до трёх названий в зависимости от того, содержится ли он в другом объекте или нет.

Название	Описание
Собственное	Имя индивидуального объекта (например <b>Valve1</b> ).
Иерархическое имя	Имя объекта в контексте его контейнера (например <b>Tank1.OutletValve</b> , где <b>Tank1</b> представляет собой имя объекта, содержащего в себе объект <b>OutletValve</b> ).
Вложенное название	Другое обозначение иерархического названия (обычно более короткое). Например <b>OutletValve</b> .

Не изменяя вложенности объекта в другой объект, непосредственно изменить его Иерархическое имя нельзя. Можно изменить его собственное или вложенное имя, при этом Иерархическое имя изменится соответствующим образом. Подробнее о принципах именования объектов в среде ArchestrA см. главу "Объекты-контейнеры".

Для изменения имени объекта нужно, чтобы он не использовался и не был захвачен.

#### Чтобы переименовать объект

1. Выделите его.
2. Выполните команду **Переименовать (Rename)** меню **Редактировать (Edit)**. Имя объекта будет отображаться в режиме изменения.
3. Измените нужным образом имя объекта и нажмите клавишу **Ввод (Enter)**.

Новое имя не должно совпадать ни с одним из имён объектов, уже существующих в Galaxy. Подробнее об именовании объектов см. параграф "Допустимые имена и символы".

---

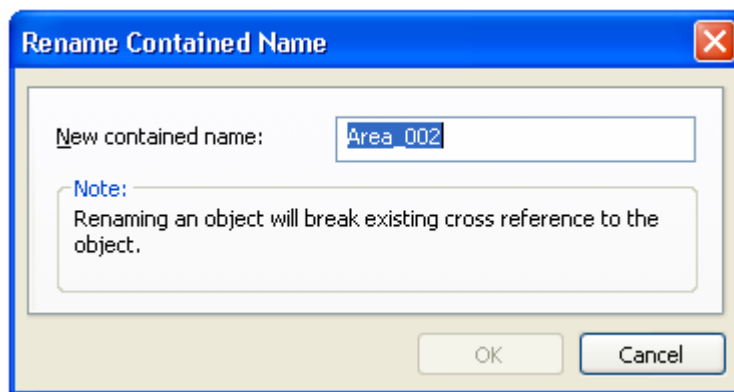
**Внимание!** Ссылки на переименованные объекты из других объектов становятся недействительными. Объекты с недопустимыми ссылками во время работы системы получают данные с плохим качеством (недоверенные).

---

Изменённое имя будет отображаться во всех ИСР, подключенных к Galaxy.

#### Чтобы изменить вложенное имя объекта

1. Выделите объект.
2. Выполните команду **Изменить вложенное имя (Rename Contained Name)** меню **Edit (Редактировать)**. Появится окно **Изменение вложенного имени (Rename Contained Name)**.



3. Измените нужным образом имя объекта и щёлкните **ОК**. Чтобы не переименовывать объект, щёлкните **Отмена (Cancel)**.

Новое вложенное имя должно быть уникальным только внутри соответствующего контейнера и удовлетворять правилам именования объектов (см. параграф "Допустимые имена и символы").

После выполнения данной операции новое вложенное имя будет отображаться во всех подключенных к Galaxy ИСР.

## Расширение функциональных возможностей объектов

Функциональные возможности объекта могут быть расширены путём указания требуемых параметров на страницах расширения, имеющих в окнах каждого редактора конфигурации. На странице **Скрипты (Scripts)** выполняется создание, редактирование и привязка скриптов, а также организация триггеров для запуска их на исполнение. На странице **Пользовательские атрибуты (UDAs)** можно определить новые атрибуты объектов и другие функции, на странице **Расширения (Extensions)** – определить параметры ввода, вывода и регистрации значений, а также генерации алармов на базе текущего набора атрибутов объекта.

Добавление новых функциональных возможностей не отменяет существующих. Подробнее см. главу "Расширение функциональных возможностей объектов".

## Шаблоны объектов

Шаблоны представляют собой уникальное средство разработки прикладной системы, играя роль её "строительных" блоков. На основе шаблонов создаются конкретные экземпляры объектов, представляющих в приложении реальные внешние устройства. Все названия шаблонов начинаются со знака доллара "\$".

Шаблон представляет собой некоторый набор атрибутов и значений по умолчанию, которые получают создаваемые экземпляры объектов. Подобная структура "шаблон–экземпляр" представляет собой основу надёжности функционирования создаваемой с помощью ИСР прикладной системы.

## Управление шаблонами

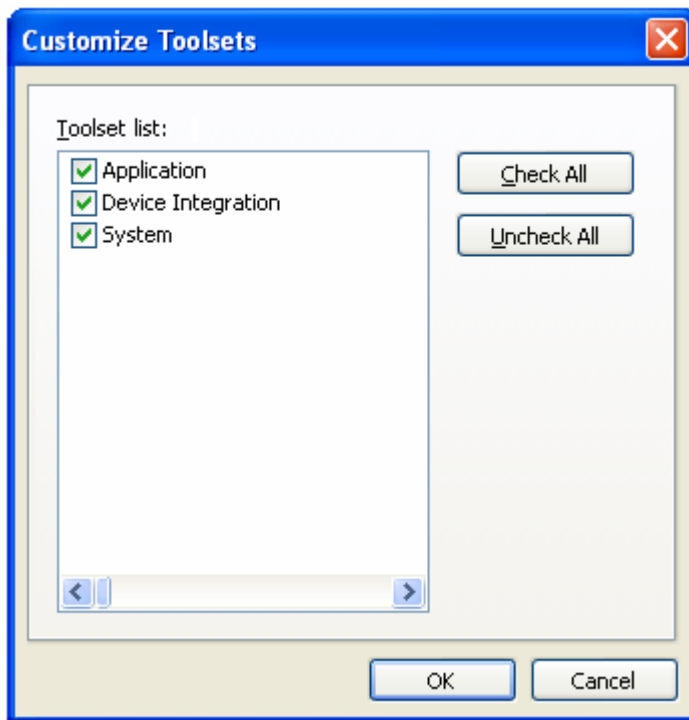
Все шаблоны организованы в виде наборов, сведения о которых выводятся в **Панели шаблонов (Template Toolbox)**.

Скрытие и отображение панели шаблонов выполняется щелчком соответствующего значка панели инструментов ИСР. Скрытие панелей

шаблонов и структуры приложений высвобождает дополнительное пространство для окон редакторов объектов.

Перемещение шаблона из одного набора в другой может быть выполнено с помощью мыши.

Для выполнения тех или иных действий с шаблонами поместите курсор мыши на пункт **Конфигурировать (Configure)** меню **Galaxy** и выполните команду **Настройка наборов инструментов (Customize Toolsets)** вложенного меню. Появится окно **Настройка наборов инструментов (Customize Toolsets)**.



В окно выведен перечень всех определённых в системе наборов шаблонов. В ИСР отображаются сведения о тех шаблонах, которые отмечены.

---

**Примечание.** Набор шаблонов, показываемых пользователю, может быть уникальным. В ИСР других пользователей могут отображаться другие группы шаблонов. Если в ИСР двух пользователей будет отображаться одно и же название набора, содержимое этих наборов будет также одинаковым.

---

Отметьте нужные группы и сбросьте лишние флажки. Отметить все группы и сбросить все флажки можно нажатием на кнопки **Отметить все (Check All)** и **Очистить всё (Uncheck All)**. Щёлкните **ОК** или кнопку **Отмена (Cancel)**, чтобы оставить содержимое панели **Панель шаблонов (Template Toolbox)** без изменений.

Чтобы увидеть содержимое шаблона, щёлкните кнопкой мыши символ "+" рядом с его именем.

---

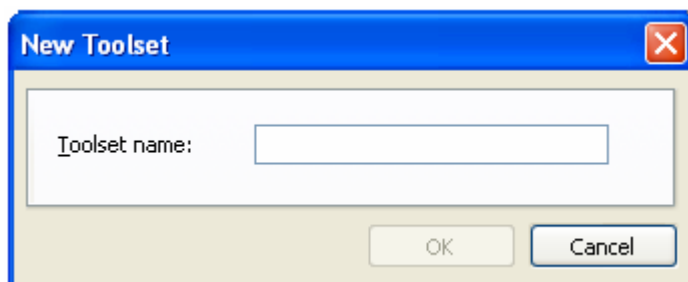
**Внимание!** Скрытие наборов в панели шаблонов не приводит к их физическому удалению из Galaxy.

---

## Создание наборов шаблонов

Чтобы создать новый набор в панели шаблонов **Template Toolbox**

1. Поместите курсор мыши на пункт **Создать (New)** меню **Galaxy** и выполните команду **Набор шаблонов (Template Toolset)**. Появится окно **Новый набор шаблонов (New Toolset)**.



2. Введите название нового набора шаблонов.
3. Щёлкните **ОК** для создания требуемого шаблона или кнопку **Отменить (Cancel)** для отмены данной операции.

В панели шаблонов появится только что созданный набор с заданным именем. После этого в него можно перемещать нужные шаблоны.

---

**Примечание.** Система проверяет допустимость указанного имени (в частности, его уникальность). При возникновении ошибки появится окно с запросом нового имени.

---

## Удаление наборов шаблонов

### Чтобы удалить набор в панели шаблонов

1. Выделите имя нужного набора шаблонов.
2. Выполните команду **Удалить (Delete)** меню **Редактировать (Edit)**.

---

**Внимание!** Перед удалением папки набора шаблонов нужно предварительно удалить все содержащиеся в нём шаблоны. Если в папке будет присутствовать хотя бы один шаблон, команда удаления будет недоступна.

---

3. Подтвердите выполнение операции, щёлкнув в открывшемся окне **Да (Yes)**. Если папку набора удалять не нужно, щёлкните **Нет (No)**.

## Создание шаблонов на основе других шаблонов

Шаблоны используются для представления целого ряда физических устройств автоматике. Например, базовый шаблон **DiscreteDevice** может использоваться для описания насосов, вентилях и прочих устройств, которые могут находиться только в двух состояниях: "включено" и "выключено". "Хорошим тоном" разработчика является предварительное определение соответствующих шаблонов перед созданием экземпляров требуемых объектов в приложении **Galaxy**.

Пусть, например, в системе используется несколько моделей насосов, выпускаемых одним и тем же производителем. Каждая модель отличается уникальными характеристиками, которые указываются в различные значения атрибутов базового шаблона **DiscreteDevice**. В данном случае "хороший тон" будет заключаться в определении производных шаблонов для каждой модели насосов на основе базового шаблона. Лучшим решением будет создать сперва иерархическую структуру производных шаблонов, а затем создать по этим производным шаблонам экземпляры нужных объектов.

### Чтобы создать новый шаблон на основе существующего

1. Выделите имя базового (исходного) шаблона в панели шаблонов или структуры приложения.
2. Поместите курсор мыши на пункт **Создать (New)** меню **Galaxy** и выполните команду **Derived Template (Производный шаблон)**. Система создаст новый производный шаблон в том же самом наборе, что и исходный, и покажет его имя в режиме изменения. Имя нового шаблона будет получено из исходного имени путём добавления некоторого числа.
3. Измените имя производного шаблона нужным образом.

**Примечание.** Если исходным является шаблон с вложенными шаблонами, в новом шаблоне будут также созданы соответствующие производные элементы. По умолчанию, структура вложенности будет сохранена вместе с именами исходных вложенных шаблонов.

Новый шаблон будет представлять собой точную копию исходного, за исключением параметров блокирования и контроля доступа. Подробнее об этих функциях см. главу "Редакторы объектов".

## Блокирование и разблокирование атрибутов шаблона

Одним из ключевых понятий взаимосвязи базовых шаблонов, производных шаблонов и экземпляров объектов являются заблокированные и разблокированные атрибуты. Обычно разработчик блокирует некоторые атрибуты объекта изменений, чтобы сохранить его целостность во время создания и конфигурирования производных шаблонов и экземпляров другими пользователями ИСР. Некоторые атрибуты объекта могут быть предназначены только для однократного определения.

Подробнее см. также главу "Редакторы объектов".

Блокирование атрибута в шаблоне подразумевает логическое распространение его значения во все производные объекты (шаблоны и экземпляры объекта). При изменении значения такого атрибута в шаблоне соответствующее изменение произойдёт во всех его дочерних шаблонах. При этом изменить значение заблокированного атрибута в дочерних шаблонах нельзя (это возможно только в исходном шаблоне).

Связь атрибутов в шаблоне и его дочерних шаблонах осуществляется по названию. После того как атрибут шаблона заблокирован, изменить его имя или удалить в дочерних шаблонах невозможно. Никакой объект нельзя модифицировать в процессе конфигурирования так, чтобы эта модификация привела к конфликту наименований атрибутов.

Возможны три типа блокировки:

Тип блокировки	Описание
Unlocked (Не заблокирован)	Данный тип блокировки действителен как для шаблонов, так и для экземпляров объектов. Значение атрибута можно как считывать, так и устанавливать. Объект имеет собственную копию значения атрибута (не передаваемую в производные объекты).
Locked (in me) (Блокирован в объекте)	Данный тип блокировки действителен только для шаблонов. Производные объекты не могут иметь значения атрибута, отличные от его значения в шаблоне (тип блокировки атрибута в них – "Блокирован в родительском объекте"). Изменение значения атрибута в шаблоне приводит к соответствующим изменениям во всех производных объектах.
Locked (in	Данный тип блокировки действителен как для шаблонов,

parent) (Блокирован в родительском объекте)	так и для экземпляров. Тип атрибута – "только чтение". Объекты не имеют собственных атрибутов, а только ссылаются на него в родительском объекте, в котором он заблокирован.
--	---

**Примечание.** Блокирование атрибута UDA во время конфигурирования превращает его в константу, и его изменение во время работы системы становится невозможным.

## Пример блокирования атрибута

1. Создайте производный шаблон на основе какого-либо шаблона и назовите его, скажем, \$Valve.
2. Запустите редактор шаблона \$Valve, установите значение одного из его атрибутов и заблокируйте его, так чтобы производные шаблоны и экземпляры объектов не могли его модифицировать. Это делается щелчком значка замка.
3. Сохраните шаблон \$Valve.
4. Создайте производный шаблон на основе шаблона \$Valve и назовите его, скажем, \$BigValve.
5. Создайте экземпляр объекта по шаблону \$Valve и назовите его Valve1.

Символ замка, отображаемый в окне редактора шаблона \$Valve рядом с именем атрибута, означает, что атрибут заблокирован (в данном объекте).

Значение этого атрибута в шаблоне \$BigValve или в объекте Valve1 изменить будет невозможно. Возможность модификации этого атрибута будет недоступна, а изображение замка (если оно есть) будет показывать, что данный атрибут заблокирован для изменений в родительском объекте. Кроме того, во всех объектах, ранее созданных на основе шаблона \$Valve, данный атрибут также будет заблокирован.

Если изменить значение заблокированного атрибута в шаблоне \$Valve, это изменение тут же будет отражено в определении шаблона \$BigValve.

## Пример разблокирования атрибута

1. Все действия будут производиться над объектами из предыдущего параграфа.
2. Разблокируйте в окне редактора шаблона \$Valve атрибут, который ранее был заблокирован.
3. Сохраните шаблон \$Valve.

Теперь значок рядом с этим атрибутом будет показывать, что он не заблокирован.

Значок рядом с этим атрибутом в шаблоне \$BigValve будет показывать, что он является заблокированным в объекте. Тип блокировки данного атрибута в экземпляре объекта Valve1 изменится на "не заблокирован", однако значок блокирования будет недоступна.

## Экземпляры объектов

Экземпляры объектов представляют собой "строительные блоки", из которых состоит Galaxy. Перед тем как использовать созданные по соответствующим базовым и производным шаблонам экземпляры объектов, их нужно разослать в места использования.

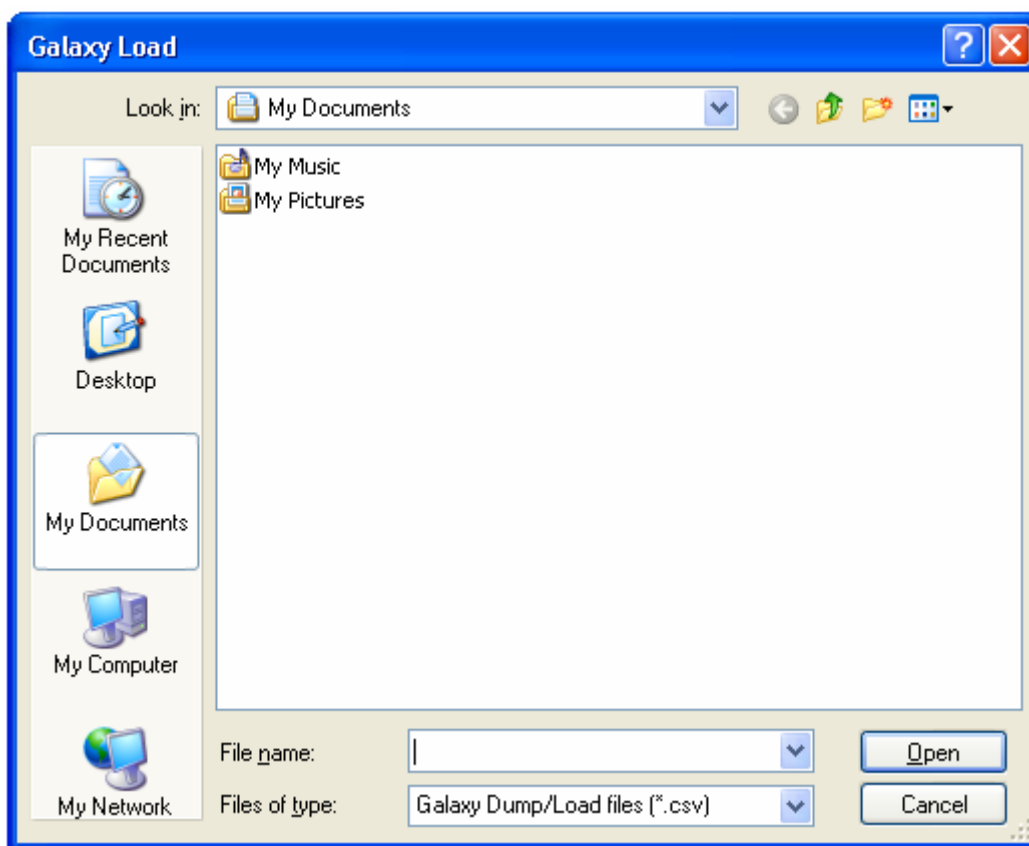
## Импортирование файлов определений объектов

Определения экземпляров объектов и значения конфигурационных параметров могут быть экспортированы из Galaxy и записаны в виде файла в формате CSV (Comma Separated Values – разделённые запятыми значения). Полученный в результате экспортирования файл с расширением .csv может быть открыт для изменения в любом текстовом редакторе или программе обработки электронных таблиц. С помощью функций редактирования, таких как копирование и вставка, можно создать любое количество определений новых объектов и затем импортировать их в любую Galaxy.

**Внимание!** В CSV-файл копируется только реальное "наполнение" Galaxy, то есть только экземпляры объектов. Записать в этот файл определения шаблонов, чтобы впоследствии импортировать их в Galaxy, невозможно. Подробнее о CSV-файлах см. параграф "Экспортирование определений объектов в файлы".

### Чтобы импортировать CSV-файл

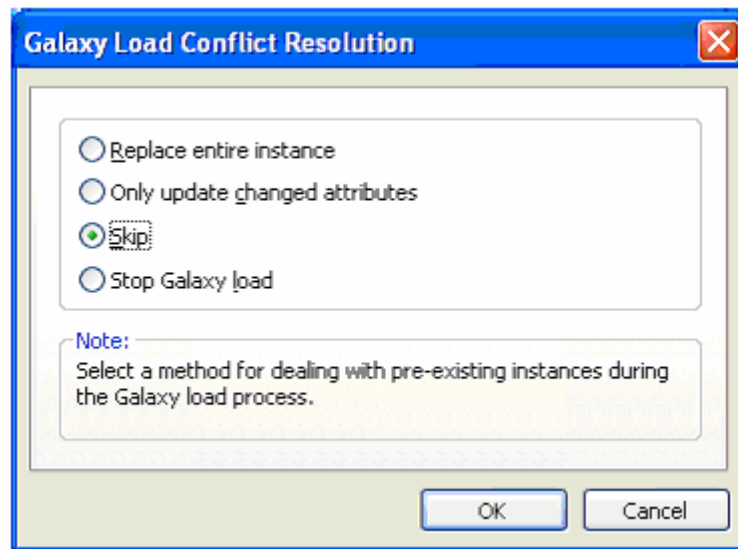
1. Поместите курсор мыши на пункт **Импортирование (Import)** меню **Galaxy** и выполните команду **Загрузка Galaxy (Galaxy Load)** вложенного меню. Появится окно **Загрузка в Galaxy (Galaxy Load)**.



2. Найдите файл с расширением .csv, в котором находится нужная для импортирования информация, и щёлкните **Открыть (Open)** для продолжения или кнопку **Отмена (Cancel)** для отмены операции импортирования.
3. Появится окно **Разрешение конфликтов во время загрузки Galaxy (Galaxy Load Conflict Resolution)**. Укажите, как должна разрешаться ситуация дублирования объектов в базе данных Galaxy, и щёлкните **ОК**. Появится окно, в котором будут отображаться результаты



импортирования объектов из указанного файла. Чтобы прекратить выполнение данной операции, щёлкните **Отмена (Cancel)**.



В окне **Разрешение конфликтов во время загрузки Galaxy (Galaxy Load Conflict Resolution)** выбирается один из следующих способов разрешения конфликтов объектов:

- **Замена объекта (Replace Entire Instance):** если в базе данных уже существует объект с таким же именем, он должен быть заменён объектом из CSV-файла.
- **Только замена значений атрибутов (Only Update Changed Attributes):** если в базе данных уже существует объект с таким же именем, изменять следует сведения об атрибутах, значения которых в файле отличны от сохранённых в базе данных.
- **Пропустить (Skip):** если в базе данных уже существует объект с таким же именем, игнорировать данные, имеющиеся в CSV-файле.
- **Прервать загрузку (Stop Galaxy Load):** если в базе данных уже существует объект с таким же именем, прекратить операцию загрузки объектов в базу данных Galaxy.

По окончании процесса загрузки все объекты, изменённые или созданные в результате выполнения этого действия, получают статус "свободные", то есть не захваченные.

---

**Внимание!** Строка комментариев в CSV-файле, созданного с помощью MS Excel, может послужить причиной создания непредусмотренного объекта со значением по умолчанию. Чтобы этого не произошло, откройте CSV-файл в Блокноте Windows и уберите из комментария символы кавычек.

---

## Экспортирование определений объектов в файлы

Определения экземпляров объектов и значения конфигурационных параметров могут быть экспортированы из Galaxy и записаны в виде файла в формате CSV (Comma Separated Values – разделённые запятыми значения).

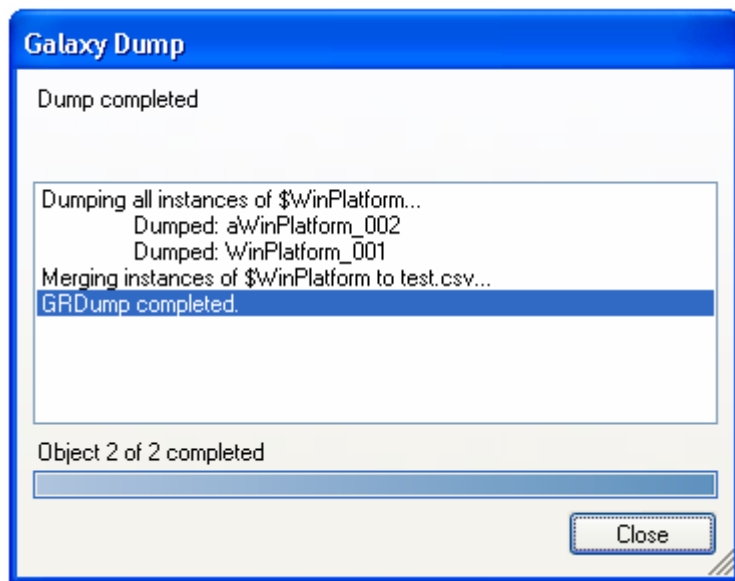
---

**Внимание!** С помощью данной функции выводятся определения только экземпляров объектов. В CSV-файл будут записаны конфигурационные параметры всех указанных объектов (текущие, если эти объекты в момент выполнения экспортирования не захвачены каким-либо пользователем, или

те, которые они имели до захвата), а также всех объектов, захваченных выдавшим команду экспортирования пользователем. В файле будут содержаться значения тех атрибутов, которые не являются заблокированными и допускают установку значений во время конфигурирования объекта (по одному значению в столбце). Значения заблокированных и вычисляемых атрибутов, а также атрибутов, получающих своё значение только во время функционирования системы, в файле не сохраняются. Кроме того, не записываются в файл значения атрибутов, не являющихся текстовыми (например атрибутов QualifiedStruct). Не сохраняется также содержимое файлов справки по объектам. Подробнее см. следующий параграф.

### Чтобы сохранить определения объектов в файле экспортирования

1. Выделите в панели структуры приложения нужный объект или объекты. Если выделить только шаблон, в файл будут экспортированы определения всех созданных по этому шаблону объектов.
2. Поместите курсор мыши на пункт **Экспортирование (Export)** меню **Galaxy** и выполните команду **Данные дампа Galaxy (Galaxy Dump)** вложенного меню. Появится окно **Данные дампа Galaxy (Galaxy Dump)**.
3. Укажите имя и каталог хранения CSV-файла, в который будут записаны определения объектов Galaxy, и щёлкните **Сохранить (Save)**. (Для отмены сохранения данных щёлкните **Отмена – Cancel**.) Появится окно, в котором будет показан ход выполнения операции сохранения.



4. По окончании процедуры экспортирования щёлкните **Закреть (Close)**.

## Структура CSV-файла экспортирования

Определения объектов Galaxy в CSV-файле сгруппированы по шаблонам, по которым эти объекты были созданы. Строки заголовка с именем шаблона содержат названия полей экземпляра. Строки комментариев организуются в файле путём указания символа точки с запятой в первой позиции строки.

При изменении файла экспортирования следует принимать во внимание следующие обстоятельства:

- В файле экспортирования имеется столбец, соответствующий атрибуту Host всех экспортируемых объектов. Для объектов Platform "Host"

всегда является именем Galaxy, из которой осуществляется вывод информации. В последующих операциях загрузки командой "Galaxy Load" эти сведения будут игнорироваться, поскольку хост-системой объекта Platform автоматически становится система, в которую этот объект загружается, независимо от того, из какой Galaxy он был экспортирован. Столбец атрибута **Host**, как и любые другие данные в файле, можно удалить. Это не окажет никакого влияния на загрузку объектов Platform, поскольку в качестве названия хост-системы они автоматически получают имя принимающей их Galaxy. Все остальные загружаемые объекты при отсутствии сведений о хосте будут помещены в папку нераспределённых объектов.

- Символы возврата каретки в связанных с экспортируемыми объектами скриптах будут в выходном CSV-файле заменены символами "\n". При редактировании файла их удалять не следует. Изменяя текст скриптов в CSV-файла, их следует указывать в качестве символа возврата каретки (во время загрузки файла они будут интерпретироваться именно таким образом). Чтобы в скрипте присутствовали символы "\n", в CSV-файле следует записать "\\n". Эта последовательность не преобразуется в символ возврата каретки во время загрузки файла.
- Особое внимание следует уделять случаям использования или удаления кавычек. Все одиночные кавычки должны указываться в файле как два идущих подряд символа кавычек, а соответствующая строка символов должна иметь как открывающие, так и закрывающие кавычки. Иными словами, в строке должно быть чётное число символов кавычек. Когда файл будет загружаться в Galaxy, дополнительные кавычки будут удалены. Например, чтобы ввести строку **3"Pipe**, в CSV-файле нужно записать **"3""Pipe"**.
- При редактировании файла экспортирования в программе MS Excel следует соблюдать требуемый формат показаний времени. Excel может изменять их формат, в результате чего они могут не загрузиться при импортировании файла. Показания времени для Galaxy должны указываться только в следующих форматах: DAYS HH:MM:SS.SSSSSS или HH:MM:SS.SSSSSS. Элемент "DAYS" в первом случае должен отделяться пробелом. Нужно учитывать, что во втором случае Excel автоматически осуществит преобразование данных в несовместимый формат. Поэтому в программе Excel показания времени всегда следует указывать в первом формате (пример допустимых значений: 0000 01:02:12.123, 1 04:03:06.12, 120 22:66:88.123456).

## Создание экземпляров объектов по шаблону

### Чтобы создать объект по шаблону

1. Выделите нужный шаблон в главном окне ИСР.
2. Поместите курсор мыши на пункт **Создать (New)** меню **Galaxy** и выполните команду **Экземпляр (Instance)** вложенного меню. В отображаемой в текущий момент структуре приложения появится новый элемент, имя которого будет показано в режиме редактирования.
3. Введите требуемое имя нового экземпляра объекта.

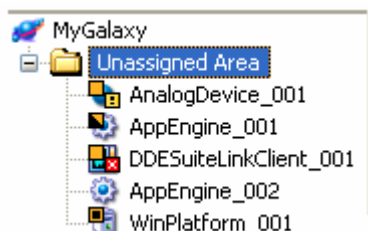
Если в системе были определены связи по умолчанию, местоположение нового объекта будет определено следующим образом:

- Объект WinPlatform будет помещён в папку зоны по умолчанию и получит параметры доступа по умолчанию.
- Объект AppEngine будет помещён в папку объекта WinPlatform в папке зоны по умолчанию.

- Объект ApplicationObject будет помещён в папку зоны по умолчанию.
- Объект DINetwok будет помещён в папку зоны по умолчанию и связан с объектом AppEngine по умолчанию.
- Объект DIDevice будет помещён в папку зоны по умолчанию, но с объектом AppEngine по умолчанию не будет связан.

## Значки состояний объектов

Ниже показан пример группирования объектов:



Как видно, изображение значков объектов изменились. Значки объектов могут содержать два типа индикаторов, обозначающих их состояние: индикатор использования (только для шаблонов) и индикатор конфигурирования (для шаблонов и экземпляров объектов).

### Индикаторы использования

	Объект не используется (см. значки объектов AnalogDevice_001 и DDESuiteLinkClient_001 в примере).
Отсутствует	Объект используется (см. значок объекта AppEngine_002)
	Объект используется с изменениями в конфигурации (см. значок объекта AppEngine_001)
	Объект используется, требуется обновление ПО (см. значок объекта WinPlatform_001)
	Отображается только в системах с резервированием. Объект AppEngine не используется, резервный компонент также не используется.
	Отображается только в системах с резервированием. Объект AppEngine используется, резервный компонент не используется.
	Отображается только в системах с резервированием. Объект AppEngine используется, резервный компонент не используется в связи с ожиданием обновления конфигурации.
	Отображается только в системах с резервированием. Объект AppEngine используется, резервный компонент не используется в связи с ожиданием обновления программного обеспечения.

Подробнее см. главу "Резервирование компонентов Archestra".

### Индикаторы конфигурирования

	Предупреждение (см. значок объекта AnalogDevice_001 в примере)
	Ошибочная конфигурация (см. значок объекта DDESuiteLinkClient_001 в примере).
Отсутствует	Ошибок в конфигурации нет (см. значок объекта AppEngine_002 в примере)

## Проверка допустимости конфигурации объектов

Каждый объект в Galaxy обладает набором конфигурационных параметров, определяющих его использование в приложении. Допустимость этого набора может проверяться как во время конфигурирования объекта, так и во время сохранения информации в базе данных Galaxy.

В проверку конфигурации объекта входят проверка допустимости значений его атрибутов, компиляция скриптов, обновление и разрешение ссылок, проверка расширений, обновление состояния и анализ прочих параметров, которые могут быть уникальными для данного объекта.

---

**Внимание!** Проверка скриптов шаблона не включает в себя разрешение указанных в них ссылок (в частности, ошибочные ссылки на несуществующие объекты UDA не определяются).

---

Для каждого из параметров, определяемых в редакторе объектов и требующих ввода строки символов или числа, имеется, как правило, соответствующий диапазон допустимых значений. При указании значений вне этого диапазона с последующей сменой страницы редактора, завершением его работы или сохранением конфигурации объекта будет выведено сообщение об ошибке ввода.

Значения некоторых параметров определяются связями с внешними компонентами, например с библиотеками скриптовых функций или со ссылками на атрибуты других объектов. Состояние этих внешних компонентов может изменяться, что в результате может приводить к блокированию некоторых функциональных возможностей данного объекта.

В частности, ошибка в конфигурации может возникать, если объект ссылается на некоторый атрибут другого объекта, а этот объект удаляется. Кроме того, конфигурирование объектов может выполняться перед импортированием соответствующих библиотек скриптовых функций. В каждом из этих случаев конфигурируемый объект будет иметь соответствующее состояние, такое как "Bad" (плохое) или "Warning" (предупреждение). Нужно вручную проверить допустимость конфигурации объекта и перевести его в состояние "Good" (годное), выполнив команду **Проверить (Validate)** меню **Объект (Object)**.

### Ручная проверка допустимости конфигурации

Чтобы выполнить проверку допустимости конфигурации одного или нескольких объектов, нужно выделить их и выполнить команду **Проверить (Validate)** меню правой кнопки или меню **Объект (Object)**. Требуемые объекты можно выделить в панели шаблонов, в панели структуры приложения или в результатах поиска в окне **Найти (Find)**.

---

**Внимание!** Вручную можно проверять конфигурацию как шаблонов, так и экземпляров объектов, но только тех, которые не захвачены каким-либо пользователем.

---

Применение функции поиска упрощает процедуру проверки конфигурации объектов. Например, с её помощью можно вывести объекты в состоянии **Егго** (ошибочное) и одной командой проверить их все.

При выполнении команды проверки появляется окно **Операции (Operations)**. Подробнее см. параграф "Панель операций".

В любой момент времени может быть запущена только одна операция проверки конфигурации. При выделении нескольких объектов они проверяются последовательно.

**Примечание.** Отменить запущенную проверку конфигурации объектов нельзя.

В ходе проверки возможно выполнение других действий, включая манипулирование объектами в проверяемом наборе. Если какой-либо из указанных объектов станет недоступным для проверки после запуска данной команды, проверка выполняться не будет. Кроме того, во время проверки конфигурации объекта другие действия выполняться с ним не будут. Сообщение о невозможности проверки будет отображено в столбце **Результаты выполнения (Command Results)** панели **Операции (Operations)**.

Чтобы выполнить проверку всех объектов Galaxy, выберите в качестве проверяемого объекта объект Galaxy.

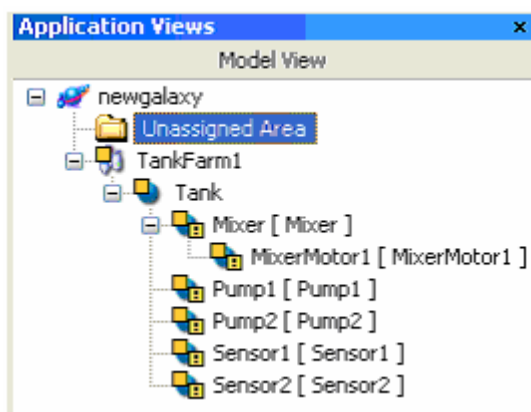
## Построение приложения

Построение приложения Galaxy осуществляется на двух уровнях: на уровне модели внешнего мира и на уровне распределения объектов.

Модель внешнего мира, представляемая общей структурой Model View в панели структуры приложения, отображает взаимосвязь процессов производственного предприятия. Как правило, процессы разделены по производственным зонам, в каждой из которых имеются различные группы технологических устройств, таких как резервуары, конвейеры, насосы, датчики и прочие механизмы.

Модель производственного предприятия в общей структуре создаётся как совокупность сложных единиц оборудования, состоящих из более простых механизмов. Например, пусть в зоне TankFarm имеется технологический резервуар. Объект, представляющий резервуар, будет входить в состав объекта, представляющего зону. В реальном резервуаре, как правило, могут использоваться и другие устройства, такие как, например, смесители, приводы смесителей, насосы и датчики. Каждое из этих устройств может представляться объектами, составляющими объект "резервуар".

Соответствующая общая структура может выглядеть следующим образом:



В данном примере "TankFarm1" представляет собой некоторую зону, являющуюся распознаваемым технологическим процессом.

"TankFarm1" в структуре вывода является экземпляром шаблона с названием "\$TankFarm1". Резервуары и прочие объекты в данной структуре также являются экземплярами шаблонов AutomationObject.

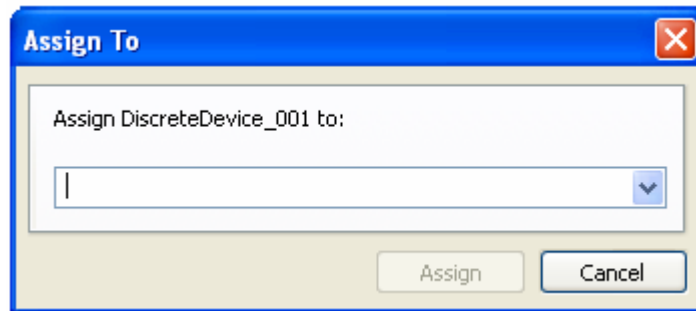
Структура использования, выводимая в панель структуры приложения, отражает распределение используемых объектов на разных компьютерах сети. Она может как соответствовать реальному расположению технологических устройств, так и отличаться. Как правило, главная причина того или иного распределения объектов по сети заключается в равномерном распределении вычислительной нагрузки по компьютерам.

## Связь объектов с хостами

**Внимание!** Некоторые связи невозможны. Допустимость связи определяется типом экземпляра объекта и типом структуры, отображаемой в панели структуры приложения. Кроме того, нельзя связать один объект с другим, если тот уже используется.

**Чтобы связать один экземпляр объекта с другим экземпляром:**

1. Выделите первый объект.
2. Выполните команду Assign To (Привязать к) меню Объект (Object). Появится окно Assign To (Привязать к).



3. Укажите имя объекта-хоста и щёлкните **Присвоить (Assign)**. Для отмены данной операции щёлкните **Отмена (Cancel)**.

**Примечание.** Более быстрым способом привязки является перетаскивание объектов с помощью мыши. Если связь объектов недопустима, будет показан универсальный символ запрета.

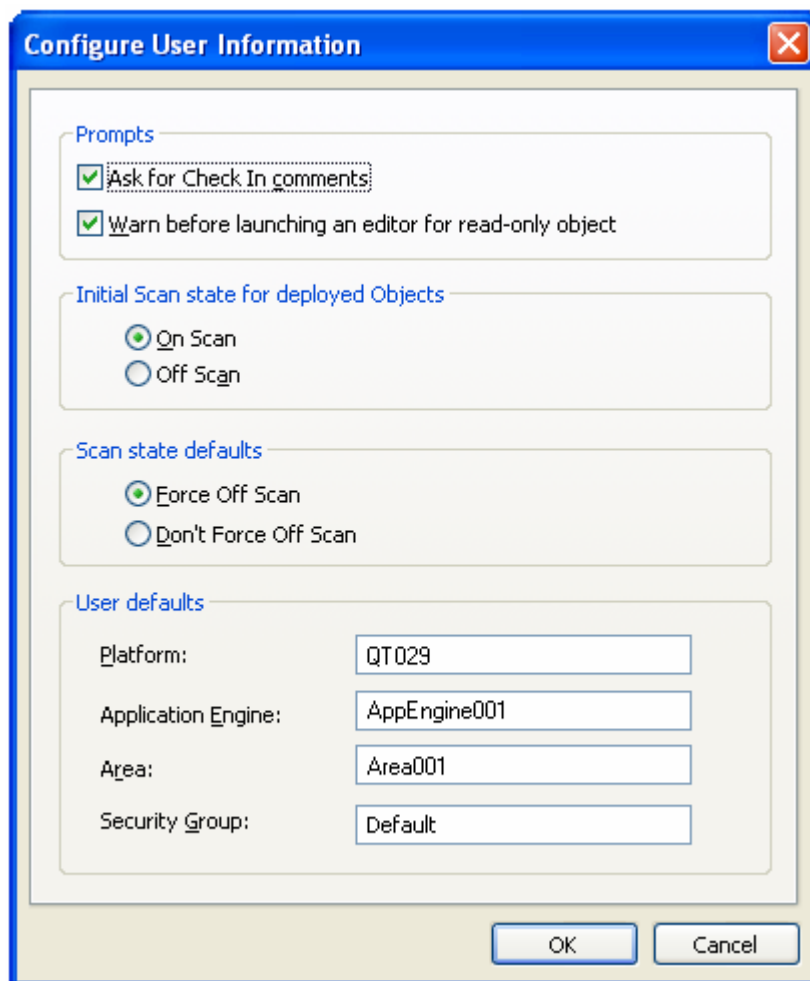
**Внимание!** Одновременно можно связать не более 30000 объектов. Попытка превысить это ограничение приводит к сообщению об ошибке и отмене выполнения указанной операции.

Связь объектов может быть выполнена также в папке **Нераспределённые хосты (Unassigned Host)**. После определения связи объектов она сразу отображается во всех подключенных к Galaxу интегрированных средах разработки.

## Связи по умолчанию

Параметры по умолчанию определяются в окне **Установка параметров пользователя (Configure User Information)** до определения экземпляров объектов по шаблонам. Это значительно упрощает процесс создания, связывания и конфигурирования объектов.

Чтобы открыть окно сведений о пользователе, выполните команду **Сведения о пользователе (User Information)** меню **Редактировать (Edit)**.



Окно содержит следующие элементы:

- **Запрашивать ввод комментариев при освобождении (Ask for Check-In Comments):** установка этого флажка означает отображение окна ввода комментариев при освобождении объекта.
- **Предупреждать о попытке редактирования объекта "только для чтения" (Warn Before Launching n Editor for Read-Only Object):** после установки этого флажка при попытке запустить редактор объекта, изменение атрибутов которого невозможно, выдаётся соответствующее предупреждение. В противном случае редактор может запускаться в режиме "только чтение", не предупреждая об этом пользователя.
- **Начальное сканирование используемых объектов (Initial Scan State For Deployed Objects):** выбор режима ("on scan" – сканировать, "off scan" – нет) сканирования используемых объектов. Изменить значение этого параметра для отдельных объектов можно в окне **Пересылка (Deploy)** во время их пересылки в место использования.
- **Сканирование по умолчанию (Scan State Defaults):** установка режима сканирования при отмене использования или повторной пересылке экземпляров объектов. Положение "Force Off Scan" отменяет сканирование целевого объекта, положение "Don't Force Off Scan" – не отменяет.
- **Параметры по умолчанию (User Defaults):** в данные поля вводятся имена объектов, которые по умолчанию должны использоваться в операциях привязки. Кроме того, в этой же панели задаётся набор прав доступа к новым объектам Galaxy по умолчанию.



---

**Примечание.** Настройка пользовательских параметров по умолчанию может быть выполнена в главном окне ИСР путём выбора нужного объекта WinPlatform, AppEngine или Aera и выполнения команды **Установить как умолчание (Set as Default)** меню **Объект (Object)**. Обозначенные таким образом объекты будут выделяться полужирным шрифтом. Объекты по умолчанию связываются с Galaxy, к которой в данный момент подключен пользователь, и они сохраняют своё состояние в последующих сеансах работы ИСР.

---

Все создаваемые впоследствии экземпляры объектов по умолчанию связываются с этими объектами и конфигурируются согласно указанным параметрам. Если параметр по умолчанию не определён, используется набор прав доступа, указанный в шаблоне.

---

**Внимание!** Указанный в данном окне в поле **Группа доступа (Security Group)** набор прав доступа отменяет соответствующую установку в определении шаблона, в окне **Параметры доступа (Configure Security)**.

---

Указанные в панелях **Начальное сканирование используемых объектов (Initial Scan State For Deployed Objects)** и **Сканирование по умолчанию (Scan State Defaults)** параметры автоматически отображаются в панелях **Уже используемые объекты (Currently Deployed Objects)** и **Начальное сканирование (Initial Scan State)** окна **Пересылка (Deploy)**. Описание этого окна см. в следующем параграфе.

---

**Примечание.** Параметры, устанавливаемые в этом окне, а также другие характеристики пользовательского интерфейса (например размеры и положение главного окна ИСР и панелей шаблонов и структуры приложения) могут быть возвращены в начальное состояние путём нажатия клавиши **Shift** в момент первоначального открытия главного окна ИСР после подключения пользователя к Galaxy и ввода регистрационных сведений.

---

## Пересылка объектов в узлы использования

Прежде чем пересылать объект на целевой компьютер, нужно выполнить следующие условия:

- Объект должен быть создан, сконфигурирован и не захвачен в Galaxy.
  - Объект должен быть связан с хостом.
  - Хост для данного объекта должен использоваться на целевом компьютере. Операция каскадной пересылки иерархической структуры объектов выполняется в правильном порядке. Хост должен быть определён до объекта, который с ним связан.
  - На целевом компьютере установлена программа начальной загрузки.
  - Состояние пересылаемого объекта в базе данных Galaxy не должно быть Еггог (ошибка).
- 

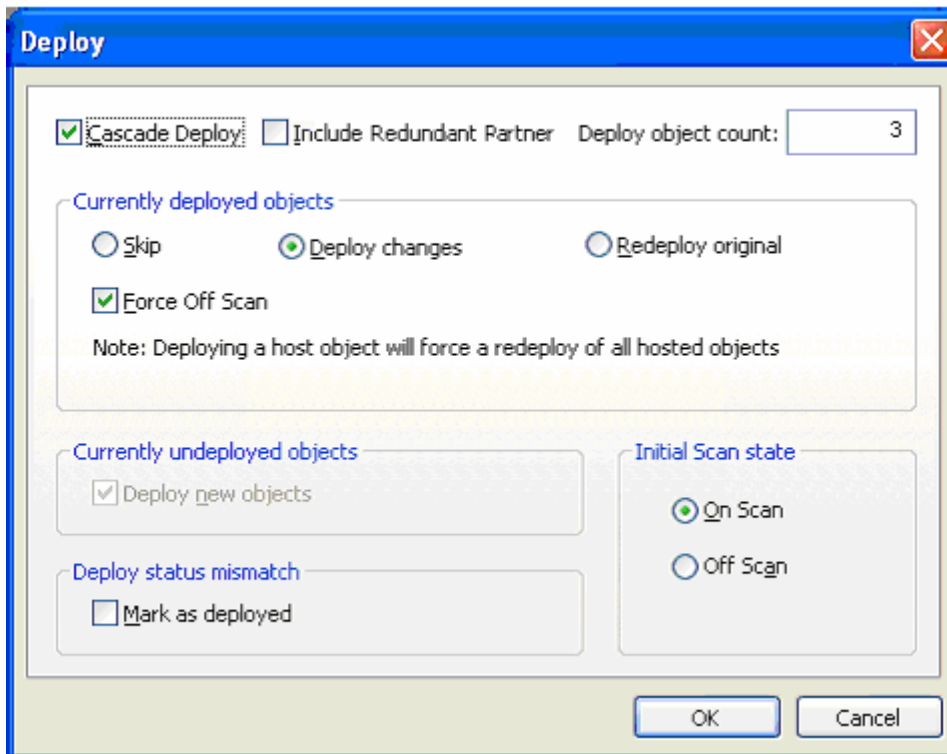
**Внимание!** Объекты DINetwork характеризуются особыми конфигурационными ограничениями (например, может ли на одной платформе WinPlatform использоваться более одного объекта или нет). ИСР ArchestrA не проверяет выполнение этих ограничений. Об указанных конфигурационных ограничениях см. справочные файлы по каждому объекту DINetwork.

---

### Чтобы переслать объект в место использования

1. Выделите объект в панели структуры приложения.

2. Выполните команду **Переслать (Deploy)** меню **Объект (Object)**. Появится окно **Пересылка (Deploy)** – см. ниже.
3. Установите параметры пересылки нужным образом и щёлкните **ОК**. Появится окно, в котором будет отображаться ход выполнения операции.



Окно содержит следующие элементы:

- **Каскадная пересылка (Cascade Deploy):** при установке данного флажка пересылается не только сам объект, но и все объекты, для которых он является хост-объектом. Если пересылаемый объект является хостом, данный флажок устанавливается автоматически. Сбросьте его, если переслать нужно только хост-объект без связанных с ним объектов.
- **Включить резервирующий объект (Include Redundant Partner):** установите этот флажок, если одновременно с данным объектом должен пересылаться резервирующий объект AppEngine. Если в текущий момент выполняется операция обновления конфигурационных параметров или программного обеспечения резервирующего объекта, данный флажок будет установлен, но недоступен. Подробнее см. главу "Резервирование компонентов систем Archestra".
- **Количество пересылаемых объектов (Deploy Object Count):** количество пересылаемых объектов. Если выбрать в панели структуры приложения несколько объектов, их количество будет показано в данном поле. Если выбрать один объект и установить флажок каскадной пересылки, в этом поле будет показано общее количество объектов в иерархической структуре связанных объектов.
- **Пропустить (Skip):** определение действий в случае, когда пересылаемые объекты уже используются. Если пересылаемый объект используется, он не изменяется в месте назначения. Если объект ещё нигде не используется, данный параметр будет недоступен.
- **Пересылать изменения (Deploy Changes):** определение действий в случае, когда пересылаемые объекты уже используются. Если

пересылаемый объект используется, система пересылает только изменённые конфигурационные параметры. Если объект ещё нигде не используется, данный параметр будет недоступен.

- **Пересылать оригинал (Redeploy Original):** определение действий в случае, когда пересылаемые объекты уже используются. Если пересылаемый объект используется, система пересылает ту же версию объекта. В частности, такой режим можно выбирать для восстановления на целевом компьютере повреждённого объекта. Если объект ещё нигде не используется, данный параметр будет недоступен.
- **Не сканировать (Force Off Scan):** определение действий в случае, когда пересылаемые объекты уже используются. Если пересылаемый объект используется, он будет исключен из процедуры сканирования, до тех пор, пока он не будет переслан на целевой компьютер. Если объект ещё нигде не используется, данный параметр будет недоступен.
- **Пересылать новые объекты (Deploy New Objects):** определение действий в случае, когда пересылаемые объекты ещё не используются. Установка этого флажка включает обычный режим пересылки объектов. Флажок будет недоступен, если объект ранее уже отсылался.
- **Отметить как отосланный (Mark as Deployed):** определение действий в случае несоответствия сведений об использовании объекта. Такое несоответствие возникает, когда объект уже отсылался на целевой узел, но в Galaxu он помечен как ещё не используемый. При установке данного флажка объект в Galaxu помечается как используемый. Сброс флажка приводит к повторной пересылке объекта на целевой узел.
- **Сканировать (On Scan):** начальное состояние сканирования для отсылаемого объекта. Состояние по умолчанию задаётся параметрами панели **Параметры по умолчанию (User Defaults)** окна **Установка параметров пользователя (Configure User Information)**. Если хост-объект для отсылаемого объекта исключён из процедуры сканирования, состояние данного переключателя игнорируется, и объект также автоматически исключается из сканирования.

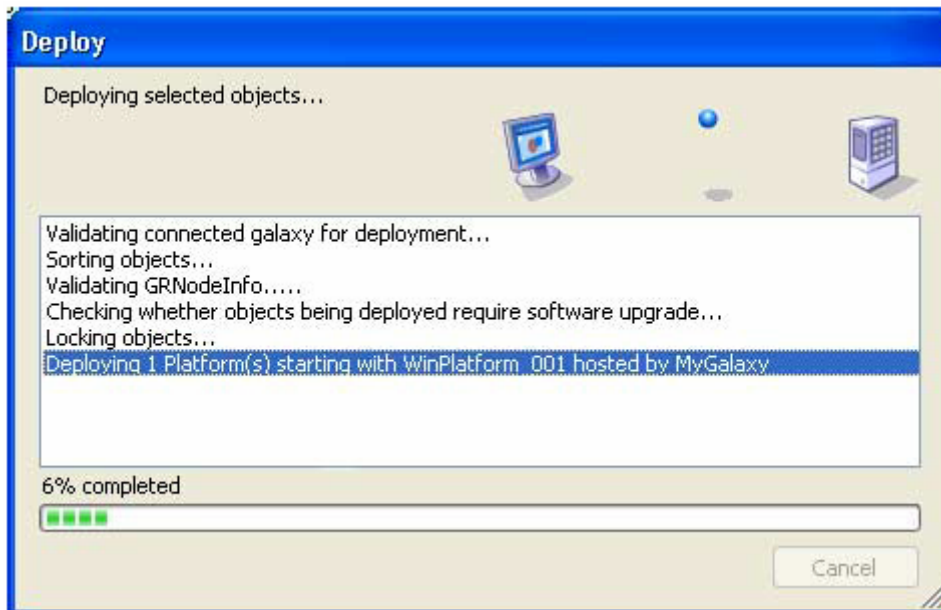
---

**Примечание.** Сканирование в объектах Area, пересылаемых в узлы с соответствующими хост-объектами AppEngine, всегда должно быть разрешено. Поскольку они являются главными поставщиками алармов, запрет сканирования приведёт к тому, что сведения об алармах и событиях не будут пересылать ожидающим их клиентам, до тех пор, пока сканирование не будет разрешено.

---

- **Не сканировать (Off Scan):** начальное состояние сканирования для отсылаемого объекта. Состояние по умолчанию задаётся параметрами панели **Параметры по умолчанию (User Defaults)** окна **Установка параметров пользователя (Configure User Information)**. Если пересылка объектов была выполнена в этом режиме, разрешить сканирование объектов и обеспечить их должное функционирование в рабочей системе можно с помощью менеджера Platform Manager, который входит в состав системной консоли управления ArchastrA.

Щёлкните **ОК** для пересылки объектов, при этом появится окно **Пересылка (Deploy)**, в котором будет отображаться ход выполнения операции.



При наличии в параметрах объекта ошибок, которые не были выявлены на этапе конфигурирования, в это окно будет выведено соответствующее сообщение. Например, пусть объём памяти на целевом компьютере недостаточен для функционирования пересылаемого объекта. Подобная ситуация, которую невозможно определить на этапе проверки конфигурации объекта, может возникнуть только во время пересылки объекта в место использования.

В окне хода пересылки отображаются как количество обработанных объектов, так и все предупреждения и сообщения об ошибках. В процессе копирования системой могут быть предприняты необходимые действия по изменению атрибутов объекта, включая атрибуты, указанные в окне пересылки.

Объекты WinPlatform являются единственными объектами, конфигурационные параметры которых указывают их планируемое местоположение. Специфическими для таких объектов проблемами пересылки являются следующие:

- Отсутствие указанного целевого компьютера в сети. Убедитесь в правильности установки параметров объекта WinPlatform и проверьте подключение целевого компьютера к сети.
- На целевом компьютере уже используется объект WinPlatform. Перед пересылкой нового объекта WinPlatform нужно отменить использование существующего.

## Повторная пересылка объектов

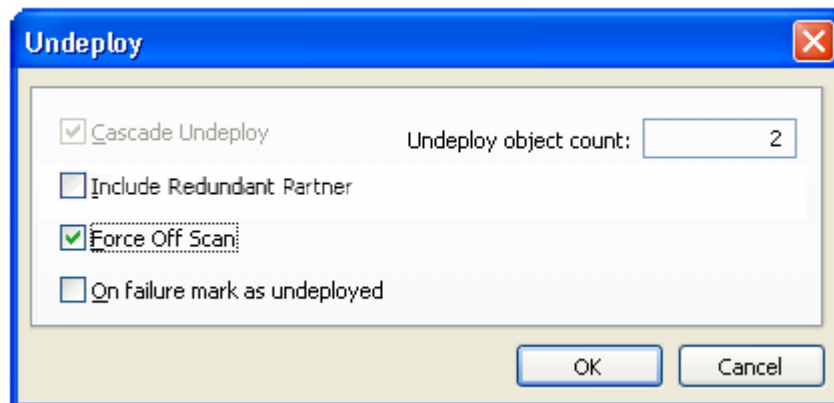
Повторная пересылка аналогична первоначальной пересылке. Для её выполнения щёлкните **Переслать (Deploy)** меню **Объект (Object)** и выполните все действия, описанные в предыдущем параграфе.

Если пересланный ранее объект имеет состояние "Pending Update", то есть его конфигурация после пересылки была изменена, и пользователь укажет необходимость рассылки этих изменений, новый объект будет помечен в Galaxy как последняя разосланная версия.

## Отмена использования объектов

**Примечание.** Невозможно отменить использование объекта, если с ним связаны другие объекты, если только не был выбран вариант каскадной отмены использования с помощью флажка **Каскадная отмена использования (Cascade Undeploy)** в окне **Undeploy**.

При выполнении команды **Отменить развёртывание (Undeploy)** меню **Объект (Object)** появится следующее окно, в котором могут быть установлены дополнительные параметры этой операции.



Окно содержит следующие элементы:

- **Каскадная отмена использования (Cascade Undeploy):** при установке данного флажка отменяется использование не только выделенного объекта, но и всех вложенных в него объектов. Доступность этого флажка определяется типом выделенных объектов.
- **Количество объектов (Undeploy Object Count):** количество объектов, использование которых прекращается. Отображаемое значение определяется как количеством одновременно выделенных в панели структуры приложения объектов, так и состоянием флажка каскадной отмены.
- **Включая резервирующий объект (Include Redundant Partner):** установите данный флажок, если одновременно с заданным объектом должно прекратиться использование резервирующего объекта AppEngine.

**Примечание.** Допускается отмена использования только объекта AppEngine, играющего в системе с резервированием роль основного (Primary), поскольку вложенные объекты станут исполняться в резервном объекте AppEngine (который станет в этом случае активным).

- **Не сканировать (Force Off Scan):** если какой-либо из объектов, использование которых прекращается, включен в процедуру сканирования и данный флажок установлен, перед отменой использования объект будет исключен из процедуры. Если сканирование целевого объекта разрешено, а флажок не установлен, операция отмены использования объекта не будет выполняться.
- **При ошибке пометить как неиспользуемый (On Failure Mark an Undeployed):** если указанный объект не будет найден, в базе данных Galaxy он будет помечен как объект, использование которого прекращено.

**Примечание.** Перед удалением и восстановлением Galaxy нужно прекратить использование всех входящих в неё объектов.

## Условия, возникающие в результате отмены использования

Прекращение использования того или иного объекта может привести к следующим результатам:

- После отмены использования объекта WinPlatform на целевом компьютере остаётся только программное обеспечение процедуры начальной загрузки. Все остальные объекты WinPlatform в системе получают соответствующие уведомления, после чего прекращают обмен данными с отменённым объектом.
- Клиенты подсистемы алармов немедленно получают извещения о прекращении использования и дальнейшей недоступности объекта зоны и удаляют ссылку на него из своих списков. Все алармы, связанные с этой зоной, также удаляются из информационных экранов (непосредственно, но не как результат вложенности в другие объекты).
- Прекращение использования объекта в состоянии обновлённой конфигурации или программного обеспечения приводит к переводу его состояния в "неиспользуемый".
- При ошибке каскадного прекращения использования одного объекта указанная операция выполняется для других объектов. Другими словами, выполнение операции не прекращается, если каскадная отмена использования одного из экземпляров объектов не удалась. Вместе с тем прекратить использование хост-объекта невозможно, если для какого-либо из его вложенных объектов эта операция завершилась с ошибкой.
- Предположим, что в активном объекте AppEngine дублированной пары существует объект ApplicationObject, в котором существуют ссылки на некоторые элементы объекта DIObject. При отмене использования объекта ApplicationObject эти элементы сразу не удаляются из набора элементов DIObject. Скорость их удаления определяется значением параметра **Максимальный период сохранения годного качества после отказа (Maximum time to maintain Good quality after failure)** (атрибут Redundancy.StandbyActivateTimeout), определяемого на странице **Резервирование (Redundancy)** редактора объектов AppEngine. Указанный параметр недействителен при отмене использования объектов ApplicationObject, входящих в состав одиночных объектов AppEngine (не имеющих резервной пары).

## ГЛАВА 3

# Редакторы объектов

Все действия по конфигурированию объектов выполняются с помощью соответствующего редактора. Открытое окно редактора объектов занимает всё пространство главного окна ИСР, не занятое панелями шаблонов и структуры приложения.

В настоящей главе описаны способы запуска редакторов и прекращения их работы, а также общие функциональные возможности конфигурирования объектов в приложении Galaxy.

### Содержание

- Запуск и прекращение работы редактора объектов
- Конфигурирование объекта в редакторе

## Запуск и прекращение работы редактора объектов

Каждый редактор объектов представляет собой набор страниц, часть которых является уникальной для данного типа объекта, а часть – одинаковой для всех объектов. Подробнее о страницах **Скрипты (Scripts)**, **Пользовательские атрибуты (UDAs)** и **Расширения (Extensions)** см. параграф "Расширение функциональных возможностей объектов" главы "Объекты".

При запуске редактора имя соответствующего объекта добавляется в список меню **Окна (Windows)**. При закрытии окна редактора имя объекта удаляется из этого списка.

### Чтобы запустить редактор объекта

1. Выделите объект.
2. Выполните команду **Открыть (Open)** меню **Galaxy**. Рядом со значком объекта появится отметка красного цвета, означающая, что объект захвачен. (Можно также выполнить команду **Открыть только для чтения – Open Read-Only**, чтобы просмотреть конфигурацию объекта в режиме "только чтение".)
3. После изменения параметров объекта выполните команду **Сохранить (Save)** или команду **Закрыть (Close)** меню **Galaxy**. В первом случае изменённые параметры записываются в базу данных Galaxy, при этом окно редактора остаётся открытым. Во втором случае окно редактора закрывается. Если объект при этом должен остаться в состоянии захвата, перед закрытием окна установите флажок **Удерживать (Keep Checked Out)**. Появится окно с предупреждением об изменении параметров объекта. Щёлкните **Да (Yes)**, чтобы сохранить изменённые данные, кнопку **Нет (No)**, – чтобы закрыть окно редактора без сохранения данных, или кнопку **Отмена (Cancel)**, – чтобы продолжить работу в редакторе.

**Примечание.** Конфигурация объекта в момент закрытия окна редактора проверяется на допустимость. При обнаружении ошибок выводится соответствующее сообщение. При этом пользователь может как отменить сохранение, так и сохранить, несмотря ни на что, текущую конфигурацию объекта. В первом случае окно редактора остаётся открытым. При сохранении конфигурации объекта в том виде, в каком она существует в текущий момент, состояние объекта изменяется на "Bad" (плохое) или "Warning" (предупреждение). Состояние объекта в базе данных Galaxy отмечается как "Good" (годное), "Warning" (предупреждение) или "Error" (ошибочное). В последнем случае использование объекта невозможно.

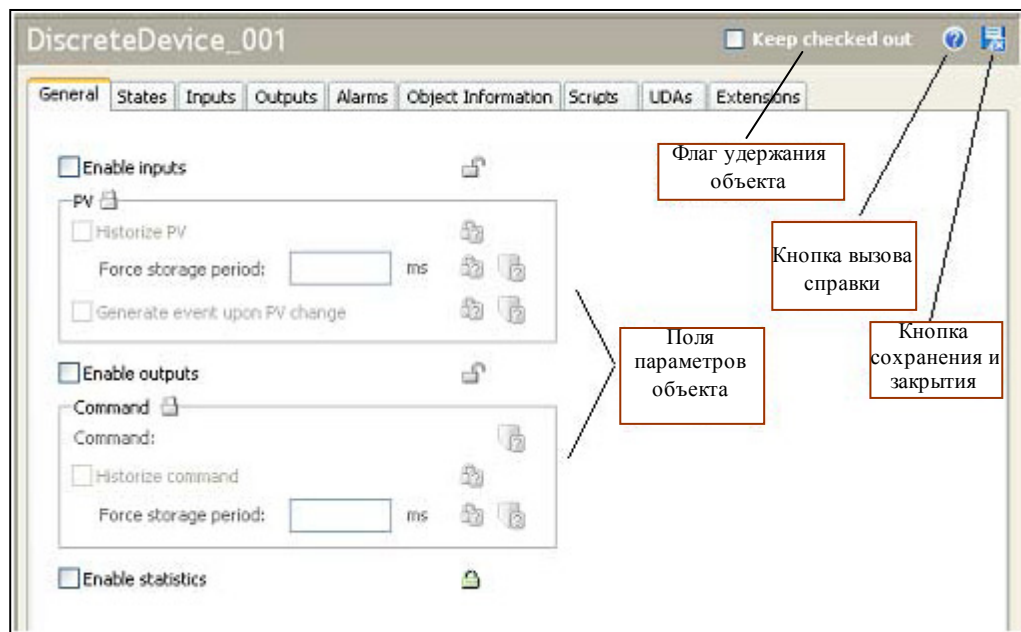
## Конфигурирование объекта в редакторе

Проверка допустимости значений каждого атрибута выполняется в момент перехода к другому полю конфигурирования объекта. Если указанное значение ошибочное, появится окно с изложением причины.

### Общий вид окна редактора объектов

Редакторы объектов имеют множество одинаковых элементов. На следующем рисунке показан примерный вид окна редактора объектов.

Проверка допустимости конфигурации объекта выполняется также при закрытии окна редактора. Если перед закрытием окна редактора установить флажок **Удерживать (Keep Checked Out)**, объект останется в состоянии захвата.



Для каждого объекта в системе имеется соответствующая справочная информация о способах его использования, конфигурирования и функционирования в рабочей системе и об атрибутах объекта. Эти сведения отображаются при нажатии кнопки с вопросительным знаком. Ниже показано примерное содержимое окна справки объекта.





Заголовок справочного файла содержит следующую информацию:

- **Имя тэга (Tagname):** имя объекта.
- **Вложенное название (Contained Name):** вложенное имя объекта (см. главу "Объекты-контейнеры").
- **Описание (Description):** краткое описание объекта.
- **Система программирования (Code Base):** версия программного кода объекта.
- **Создан на основе (Derived From):** название прямого "родителя" данного объекта (например производного шаблона).
- **Версия объекта (Object Version):** версия конфигурации объекта.
- **Порядок обработки (Process Order):** момент обслуживания объекта в рабочей системе (none – нет, before – перед, after – после) по отношению к объекту, указанному в поле **Соседний объект (Relative Object)**.
- **Соседний объект (Relative Object):** объект, который обслуживается перед данным объектом или после него (определяется значением в поле **Порядок обработки – Process Order**).

Если файл справочной информации для объекта не существует, в окне будет показан путь к каталогу, в котором он должен находиться. Подобная ситуация может возникнуть в результате удаления справочного файла из

базы данных Galaxy. Нужно создать в указанном каталоге файл Help.HTM или повторно импортировать определение данного объекта.

При помещении указателя мыши на элемент окна редактора будет выведено имя соответствующего атрибута.





Чтобы сохранить изменения конфигурации объекта, щёлкните **Сохранить и закрыть (Save and Close)**. Объект при этом освобождается, если только перед закрытием окна не был установлен флажок **Удерживать (Keep Checked Out)**. После освобождения объекта его конфигурационные параметры становятся доступными всем пользователям.

## Блокировка

Взаимодействие пользователя с объектом осуществляется путём изменения его атрибутов. При блокировании атрибута его дальнейшее изменение становится невозможным. Некоторые атрибуты могут блокироваться разработчиком, для того чтобы они оставались неизменными во всех экземплярах объекта. Некоторые являются незаблокированными и могут блокироваться пользователем, что позволяет предотвращать их изменение в порождённых объектах. Для каждого параметра в любой момент времени может быть указан только один вид блокировки.

Если атрибут заблокирован, для него будет показан соответствующий значок. Если данная настройка доступна, изменить тип её блокировки можно с помощью мыши, щёлкнув данный значок.

В следующей таблице перечислены возможные типы блокировок:








Значок блокировки	Значение	Описание
	Locked (in me) – Блокирован (во мне)	Помеченный так параметр заблокирован в текущем объекте и доступен (только для шаблонов).
	Locked (in parent) (Блокирован в родительском объекте)	Помеченный так параметр заблокирован в родительском объекте и недоступен в дочернем объекте.
	Unlocked (Не заблокирован)	Помеченный так параметр не заблокирован и доступен. Только для незаблокированных параметров в шаблонах.
	Indeterminate (Неопределим)	Этот значок относится только к группе параметров. Неопределимое состояние означает, что различные параметры в группе находятся в разных состояниях блокировки.
	Undefined (Не определено)	Прежде чем блокирование может быть применено к данному атрибуту, нужно разрешение доступа к другому параметру.

## Контроль доступа

Для каждого параметра, или атрибута в окне редактора может быть установлены различные виды доступа, определяющие доступ к ним в рабочей системе. Для каждого атрибута в любой момент времени может быть указан только один вид доступа.

**Внимание!** Настройка видов доступа возможна только после выбора одного из трёх режимов идентификации пользователя. Подробнее см. главу "Контроль доступа".

Вид доступа отображается в виде соответствующего значка. Если изменение вида доступа поддерживается, пользователь может задать одно из следующих состояний:

Значок	Значение
 Free access (Свободный доступ)	Изменить значение этого атрибута может любой пользователь, даже не имеющий явно определённых прав доступа к объекту. Это позволяет производить над объектом срочные операции, не терпящие задержек (например для остановки разбалансированного процесса).
 Operate (Использование)	Пользователи с полномочиями Operate могут выполнять определённые стандартные операции, в число которых входит установка значений таких атрибутов ПИД-объекта, как Уставка (Setpoint), Выход (Output) и Режим управления (Control Mode), или атрибута Команда (Command) объекта Логическое устройство (Discrete Device).
 Secured Write (Гарантированная запись)	Чтобы изменить значение атрибута, пользователь с полномочиями Operate должен выполнить повторный ввод своих регистрационных данных. Подобный уровень защиты обычно задаётся для объектов, имеющих в системе высокую степень важности.
 Verified Write (Запись под контролем)	Чтобы изменить значение атрибута, требуется повторная регистрация уже зарегистрированного пользователя, а также регистрация второго (контролирующего) пользователя. Подобный уровень защиты обычно определяется для объектов, имеющих в системе высокую степень важности.
 Tune (Настройка)	Настраивать значение этого атрибута в рабочей системе могут конечные пользователи с полномочиями Tune Operational (операции настройки). Примерами подобной настройки являются изменение контрольных точек алармов и изменение чувствительности ПИД-регуляторов.
 Configure (Конфигурирование)	Настраивать значение этого атрибута в рабочей системе могут конечные пользователи с полномочиями Configure Operational (операции конфигурирования). Нужно, чтобы предварительно они отменяли сканирование объекта. Установка значения этого атрибута рассматривается как одно из существенных изменений конфигурации (это, например, смена регистра ПЛК, передающего данные на вход объекта Discrete Device – Логическое устройство).
 Read Only (Только чтение)	Независимо от имеющихся полномочий, во время работы системы пользователи могут только просматривать значение данного атрибута. Изменение значения атрибута во время работы системы невозможно.

Значок типа доступа может быть окрашен в серый цвет, что означает, что атрибут заблокирован в родительском объекте и не может быть изменён, или же что он требует разрешения группового атрибута.

## Групповое блокирование и управление доступом

Групповое блокирование и управление доступом позволяет устанавливать соответствующие параметры для всех элементов группы.

Групповой значок обычно отражает установку для всех элементов группы. Но если установка параметра хотя бы у одного элемента отличается от остальных, тип групповой установки будет Неопределимый (Indeterminate).

Групповые значки по значению совпадают со значками отдельных объектов.

## Функциональные клавиши

Переход в окне редактора от одного параметра к другому (как правило, от самого верхнего левого к самому нижнему правому) может быть выполнен нажатием клавиши табуляции **ТАВ**. При этом переход к заблокированным параметрам не осуществляется. При редактировании текста в текстовом поле нажатие данной клавиши приводит к вставке в текст символа табуляции.

При нажатии клавиш **HOME** и **END** курсор устанавливается в начало и в конец поля ввода данных соответственно. Перемещение курсора в текстовых полях и в полях со списком может быть выполнено нажатием клавиш управления курсором.

Обработка данных в поле ввода выполняется после перехода к другому параметру в окне редактора (например при нажатии клавиш **ENTER** или **ТАВ**, также как и при щелчке кнопкой мыши вне поля). Одновременно выполняется проверка допустимости введенного значения.

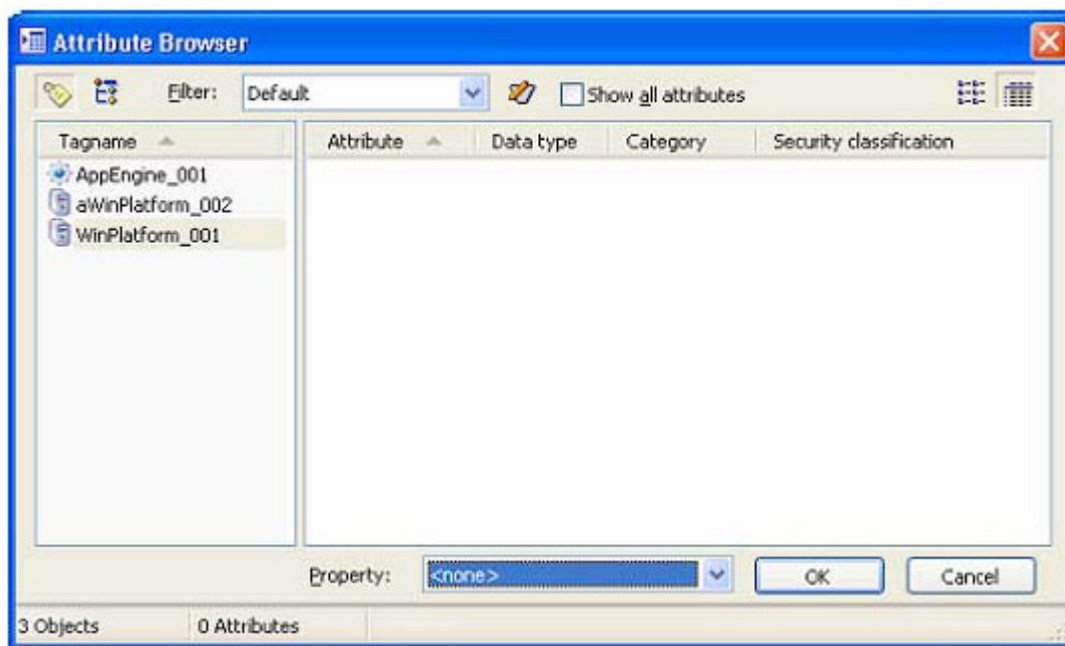
Нажатие клавиши **ESC** отменяет текущее редактирование значения поля и восстанавливает его предыдущее состояние.

После выделения флажка или переключателя изменить его состояние можно нажатием клавиши пробела.

## Просмотр объектов с помощью браузера атрибутов

Браузер атрибутов представляет собой одно из средств Archestra, с помощью которого осуществляется поиск нужного атрибута или свойства для ссылки на него в конфигурации другого объекта.

В некоторые поля окна редактора объектов, таких как **Ссылка (Reference)**, нужно вводить ссылки на другие объекты. Быстро найти требуемый объект можно, нажав кнопку **Браузер атрибутов (Attribute Browser)**. Появится окно следующего вида:





Подробнее см. главу "Расширение функциональных возможностей объектов".


## Интерфейс браузера атрибутов

Окно браузера атрибутов состоит из следующих четырёх частей: верхней панели параметров, левой панели, правой панели и панели выбора свойств.

### Панель параметров

Выбор типа отображения в левой панели (имён объектов или иерархических названий) осуществляется нажатием одной из следующих

кнопок:  – Имена Тэгов (Tagname) или  – Иерархия Имен (Hierarchical Name).

Ограничить количество отображаемых элементов можно, создав фильтр поиска с помощью кнопки  – Создать фильтр (Edit Filter).

Фильтры могут применяться как к именам объектов, так и к названиям атрибутов. После создания фильтра его имя появляется в списке **Фильтр (Filter)**. Фильтр "По умолчанию" (Default) обеспечивает вывод списка всех объектов и атрибутов. Изменять его нельзя. Подробнее см. параграф "Создание фильтров".

По умолчанию браузер отображает только те атрибуты, которые помечены как часто используемые. При просмотре объекта впервые правая панель может оказаться пустой. Для вывода списка атрибутов такого объекта щёлкните **Показать все атрибуты (Show All Attributes)**.

### Левая панель

В левой панели отображается список, который состоит из захваченных пользователем объектов, а также из "не захваченных" версий всех остальных объектов.

---

**Внимание!** В случае дублированной пары объектов в окне браузера отображаются сведения только об основном объекте AppEngine и его атрибутах. Сведения о резервирующем объекте AppEngine не выводятся. Подробнее см. главу "Резервирование компонентов систем ArchestrA".

---

## Правая панель

В правой панели отображается список атрибутов объекта, выделенного в левой панели. Таблица атрибутов содержит следующие столбцы:

- **Атрибут (Attribute):** имя атрибута вместе со значком соответствующей группы типа данных.
- **Тип данных (Data Type):** тип значений атрибута: String (строковый), Integer (целый), Boolean (логический), Reference (ссылочный), Time (временное), InternationalizedString (международная строка) и CustomEnum (перечисление).
- **Категория (Category):** информация о том, разрешены ли установка и чтение значений атрибута во время конфигурирования системы и работы с ней.
- **Тип доступа (Security Classification):** тип доступа к атрибуту во время работы системы: Tune (настройка), Operate (работа), Free Access (свободный доступ), Secured Write (защищённая запись), Verified Write (проверяемая запись), View Only (только просмотр). Данный параметр задаёт вид доступа к атрибуту и его значениям. Если у атрибута тип доступа не определён, соответствующая ячейка этого столбца будет не заполнена.

## Панель выбора свойств

После выбора атрибута можно указать точно, какое из его свойств должно быть использовано в ссылающемся на него объекте. Список возможных свойств определяется типом выбранного атрибута, и он может включать следующие элементы:

- `<none>` (нет): означает автоматический выбор свойства Value (Значение) атрибута.
- Категория (Category): определяет, когда значения атрибута доступны (во время конфигурирования или работы системы), какие пользователи могут устанавливать его значения, является ли данный атрибут блокируемым или неблокируемым.
- Размерность (Dimension1), только для массивов: возвращает размерность массива (если данный атрибут является массивом);
- Заблокирован (Locked): показывает, заблокирован ли в текущий момент данных атрибут. Допустимые значения: Unlocked (не заблокирован), LockedInMe (блокирован в текущем) и LockedInParent (блокирован в родительском).
- Качество (Quality): определение качества данных атрибута в соответствии со спецификацией OPC Draft 3.0. Качество по OPC в среде ArchestrA хранится и передаётся в виде 16-разрядной величины. Качество по OPC атрибута хранится как текущее качество, информация о котором может быть записана в архив и передана клиентам.
- Тип доступа (SecurityClassification): показывает, какие полномочия имеет пользователь для выполнения тех или иных действий над значениями атрибута во время работы приложения ArchestrA. Имеет смысл только для тех атрибутов, установка значений которых возможна во время работы приложения.
- Тип (Type): тип значений атрибута: Integer (целый), Boolean (логический), Float (вещественный), Double (двойной вещественный), String (строковый), InternationalizedString (международная строка), Time (временной), ElapsedTime (прошедшее время), ReferenceType (ссылочный тип), CategorizedStatusType (тип категорийного

состояния), DataTypeEnum (перечисление типа данных), SecurityClassificationEnum (перечисление классификации доступа), DataQualityType (тип качества данных), CustomEnum (пользовательское перечисление), CustomStruct (пользовательское структура).

- Значение (Value): первичное значение атрибута.

В некоторых случаях в список свойств входит набор чисел. Эти числа обозначают номер разряда в целом значении свойства Значение (Value). Допустимые спецификаторы разрядов:

- .01 (наименьший значащий разряд).
- .01 .02 .03 .04 .05 .06 .07 .08 .09 .10 .11 .12 .13 .14 .15 .16 .17 .18 .19 .20 .21 .22 .23 .24 .25 .26 .27 .28 .29 .30.
- .31 (наибольший значащий разряд).

---

**Внимание!** Спецификаторы разрядов нельзя использовать для ссылки на массивы целых чисел. Несмотря на то, что обращение к разрядам поддерживается только для величин целого типа, они могут быть введены и для других типов (никакого предупреждения в процессе конфигурирования не выдаётся), но во время работы приложения их использование приведёт к возникновению ошибки.

---

Явно указывать свойство атрибута не обязательно. Если выделить в браузере атрибут, ничего не выбирая в списке его свойств, по умолчанию будет возвращено свойство Значение (Value).

## Использование браузера атрибутов

Если в поле параметра в окне редактора объектов уже имеется ссылка, браузер атрибутов отображает её или расширяет до ближайшего определённого в Galaxu совпадающего сочетания "объект.атрибут.свойство".

Если выделить в редакторе скриптов какой-либо текст, он будет использоваться браузером атрибутов как основа дальнейшего поиска соответствующего атрибута.

После того как требуемые атрибут и свойство определены, щёлкните **ОК** для вставки их в окно редактора объектов, или щёлкните **Отмена (Cancel)** для возврата без вставки. При работе с редактором скриптов найденная ссылка будет вставлена в позиции курсора вместо выделенного текста.


---

**Внимание!** Список объектов в браузере атрибутов отображается по принципу "как в последний раз" (в виде списка собственных или иерархических имён). Выбор частично введённой ссылки в редакторе объекта, не соответствующей предыдущему типу списка браузера, не приводит к изменению режима отображения. Иными словами, если при последнем обращении к браузеру он отображал список собственных имён объектов, – при очередном его вызове список будет отображаться так же, независимо от способа указания ссылки в окне редактора объектов.


---

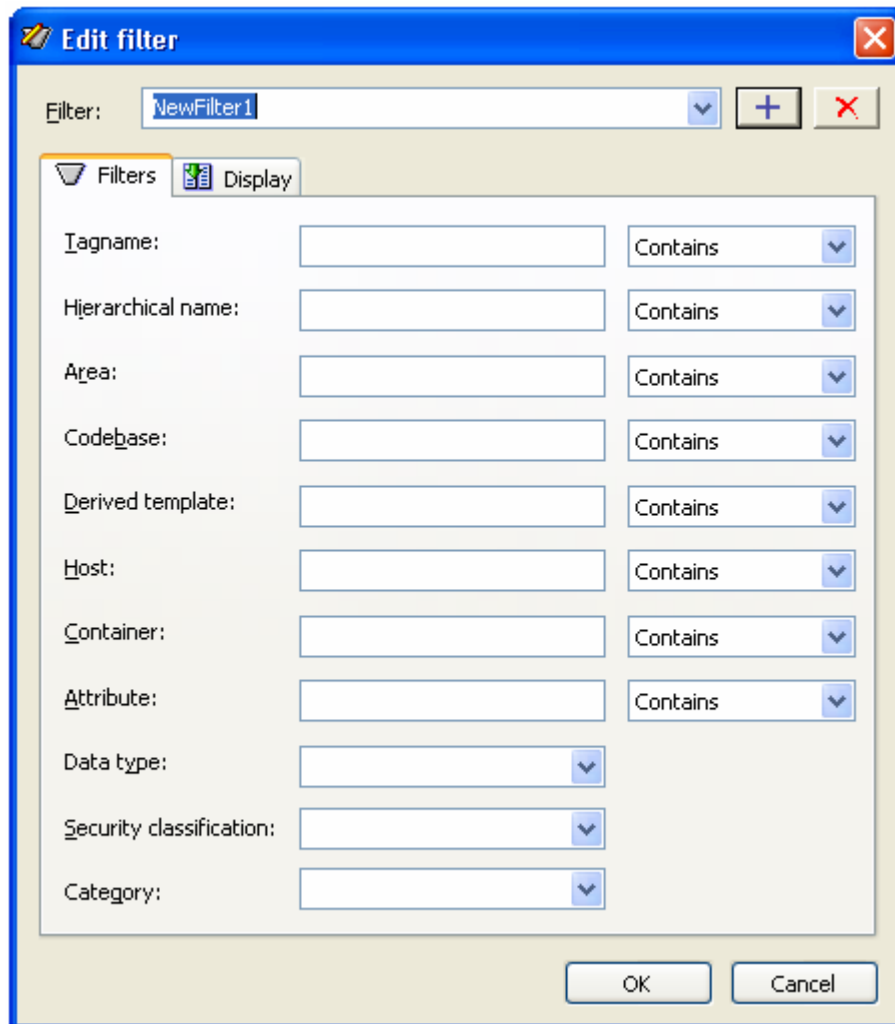
## Создание фильтров

Объём данных в окне браузера атрибутов может быть очень большим. Сократить его можно путём определения фильтра поиска.

Чтобы создать фильтр, щёлкните кнопку  – **Редактировать фильтр (Edit Filter)**. Фильтры могут применяться к именам, связям, вложению и типу образования объектов; к названиям атрибутов, к их типам, категориям и видам доступа; а также к полям объектов и атрибутов, отображаемым в браузере объектов.

Первоначально в окне **Редактировать фильтр (Edit Filter)** отображается фильтр "По умолчанию" (Default), который определяет вывод списка всех объектов и их атрибутов. Чтобы определить новый фильтр, щёлкните

кнопку со знаком плюса . В поле **Фильтр (Filter)** появится генерируемое системой имя нового фильтра. Измените его нужным образом.



Окно правки фильтра содержит две страницы, на которых настраиваются различные типы фильтров. На странице **Фильтры (Filters)** устанавливаются параметры, позволяющие уменьшать количество отображаемых объектов и атрибутов, на странице **Отображение (Display)** – параметры, позволяющие уменьшать количество полей для каждого объекта и атрибута.

Страница поиска имеет восемь элементов, для которых можно указать критерии выбора (по умолчанию будет выбран критерий **Содержит – Contains**). Возможны следующие критерии:

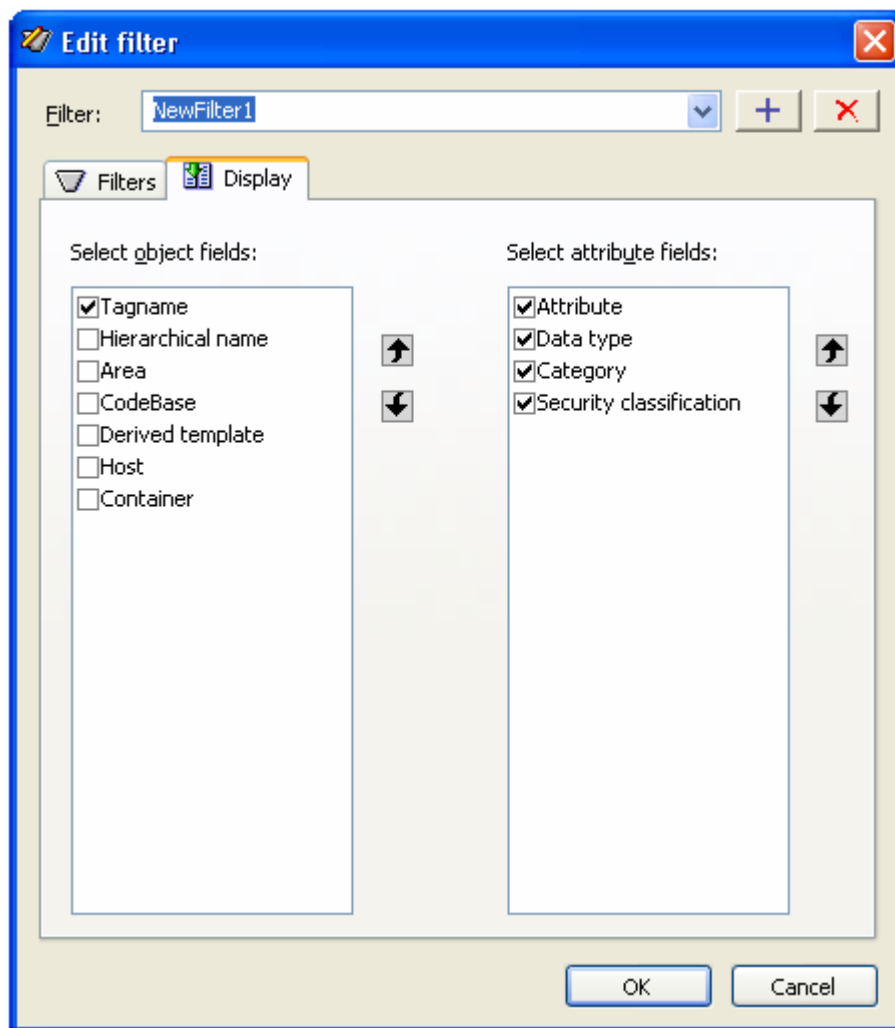
- Содержит (Contains).
- Равно (Exact Match).
- Начинается строкой (Starts With).
- Заканчивается строкой (Ends With).

Кроме того, имеются три поля настройки дополнительных критериев поиска:



<b>Data Type (Тип данных)</b>	<b>Security Classification (Тип доступа)</b>	<b>Category (Категория)</b>
Boolean	Configure	Calculated
CustomEnum	Free Access	CalculatedRetentive
CustomStruct	Operate	Constant
DataType	Secured Write	PackageOnly
Double	Tune	PackageOnly_Lockable
ElapsedTime	Verified Write	SystemInternal
Float	View Only	SystemInternal_Browsable
Integer		SystemSetsOnly
InternationalizedString		SystemWriteable
Quality		Writeable_C_Lockable
Reference		Writeable_S
SecurityClassification		Writeable_U
Status		Writeable_UC
String		Writeable_UC_Lockable
Time		Writeable_US
		Writeable_USC
		Writeable_USC_Lockable

Выберите нужные критерии поиска и щёлкните **OK**.



На странице **Отображение (Display)** показаны два списка элементов отображения: один, начинающийся со столбца **Имя тэга (Tagname)**, – для списка объектов в левой панели браузера атрибутов, второй, начинающийся со столбца **Атрибут (Attribute)**, – для атрибутов в правой панели. Отметьте элементы, которые должны отображаться в браузере атрибутов. Изменить порядок отображения можно с помощью кнопок со стрелками, расположенных справа от соответствующего списка.

---

**Примечание.** Элементы **Имя тэга (Tagname)** и **Атрибут (Attribute)** всегда будут отмечены для отображения, и они всегда будут стоять на первом месте соответствующего списка.

---

Отметьте элементы, которые должны отображаться в окне браузера атрибутов, и щёлкните **ОК**.

После того как фильтр будет определён, его имя появится в списке поля **Фильтр (Filter)** окна браузера атрибутов.

Чтобы вывести список всех объектов и атрибутов без указания критериев поиска, выберите в этом списке пункт **По умолчанию (Default)**.

## Страница сведений об объекте

Страница **Сведения об объекте (Object Information)** одинакова для всех редакторов объектов.

Platform History Scheduler History Engine History **Object Information** Scripts UDAs Extensions

Description: The Platform represents a computer in the automation application.

---

Hierarchical name: WinPlatform\_001

Container: N/A

Code base: ArchestraA.WinPlatform.1

Derived from: \$WinPlatform

Host: MyGalaxy

Area: N/A

Security group: Default

---

Click the button to add help for this object. [Add object help](#)

Эта страница содержит следующие элементы:

- **Описание (Description):** краткое описание объекта.
- **Иерархическое имя (Hierarchical Name):** полное уточнённое имя вложенного объекта, включая имя объекта-контейнера.
- **Контейнер (Container):** имя объекта, содержащего данный объект (если имеется).
- **Система программирования (Code Base):** версия программного кода объекта.
- **Создан на основе (Derived From):** имя непосредственного "родителя" данного объекта (например производного шаблона).
- **Хост-объект (Host):** имя объекта, с которым связан данный объект (например WinPlatform, с которой связан объект AppEngine).
- **Зона (Area):** объект, представляющий логическую группу, в которую входит данный объект.
- **Тип доступа (Security Group):** имя группы прав доступа к данному объекту (см. также главу "Контроль доступа").
- **Создать справку об объекте (Add Object Help):** при нажатии этой кнопки появляется окно со справочной информацией о шаблоне, по которому был создан текущий объект. На основе этой информации можно создать справочный файл, содержащий сведения об объекте. Указанные сведения будут отображаться на экране пользователей при выполнении команды **Справка об объекте (Object Help)** меню **Справка (Help)** или меню правой кнопки.

**Внимание!** Приложение Microsoft Word не поддерживается как текстовый редактор справочной информации. Если при нажатии кнопки **Создать**

**справку об объекте (Add Object Help)** появляется окно приложения Word, нужно изменить тип редактора HTML-файлов на странице **Типы файлов (File Types)** окна **Свойства папки (Folder Options)** Проводника Windows.

---

На странице сведений некоторых объектов может отображаться дополнительная группа параметров с названием **Порядок обслуживания (Execution Order)**. С их помощью определяется порядок обслуживания текущего объекта в зависимости от обслуживания другого объекта: непосредственно перед ним или после него. В данной группе имеются два поля: поле **Порядок обработки (Process Order)** с параметрами **Перед (Before)** и поле **После (After)** и **Соседний объект (Relative Object)**, обслуживание которого является определяющим моментом обслуживания данного объекта. В поле **Соседний объект (Relative Object)** имени объекта обязательно должен предшествовать символ "@": если имя объекта вводится вручную, этот знак также должен указываться, если выбор объекта осуществляется с помощью браузера атрибутов, этот знак добавляется автоматически.

---

**Внимание!** Вставляя в текст справки какие-либо изображения, нужно помещать графические файлы в соответствующую папку компьютера Репозитория Galaxu и указывать в справочном HTML-файле относительный путь доступа к этим файлам. Каталог справочного HELP-файла каждого объекта уникален и определяется каталогом, указанным при установке Репозитория Galaxu. Путь доступа к справочному файлу объекта записывается как \\<Каталог установки>\Framework\FileRepository\<Имя Galaxu>\Objects\<Имя объекта>\Help\1033. Каталогом установки по умолчанию является папка \\Program Files\Archestra\. Файлы графических изображений можно записать как в папку \1033, так и во вложенную в неё папку.

---

# Архивирование

В качестве системы архивирования сервера промышленных приложений Industrial Application Server используется архиватор InSQL (InSQL) компании Wonderware. В базе данных InSQL могут храниться значения атрибутов любых объектов ArchestrA.

В настоящей главе описываются способы конфигурирования объектов с целью сохранения хронологических значений в базе данных архиватора InSQL.

Подробнее об архивировании накапливаемой информации см. в Руководстве по разработке проектов FactorySuite A<sup>2</sup>.

## Содержание

- Архиватор InSQL
- Настройка объектов WinPlatform и AppEngine для архивирования данных
- Настройка остальных объектов для архивирования данных

## Архиватор InSQL

Установка архиватора InSQL должна выполняться с соблюдением следующих условий:

- Архиватор должен быть установлен на компьютере, отличном от компьютера Galaxy, но входящем в локальную сеть.
- На этом же компьютере не должно быть никаких установленных компонентов ArchestrA.
- Архиватор может получать хронологические данные только от одной Galaxy.

Архиватор InSQL поддерживает определения качества, действительные для IAS (спецификация OPC Draft 3.0). В определённых условиях (например при отключении сети из-за предполагаемого простоя) он также способен создавать дополнительные записи VTQ с модификацией качества. Кроме того, все сведения, получаемые от компонентов сервера IAS, при сохранении в базе данных InSQL дополняются двумя не соответствующими требованиям спецификации OPC полями (Quality и QualityDetail).

Подробнее о качестве данных в среде ArchestrA см. параграф "Качество данных".

---

**Внимание!** Сервер IAS обменивается данными с узлом InSQL через интерфейс, называемый MDAS (Manual Data Acquisition Service – Служба ручного накопления данных). MDAS, в свою очередь, опирается на модель DCOM (Distributed COM), согласно которой в узле InSQL должен быть доступен порт TCP/UDP с номером 135. Если этот порт недоступен, сервер

IAS не сможет сохранять данные в архиве InSQL. Одной из возможных причин, по которой порт 135 может быть недоступен, может являться наличие маршрутизатора между узлом сервера IAS и узлом InSQL, блокирующего обращение к этому порту. Кроме того, порт может быть недоступен из-за того, что в каком-либо из узлов применение DCOM запрещено. Чтобы сервер IAS и архиватор InSQL могли взаимодействовать, при конфигурировании сервера нужно разрешить использование DCOM.

---

## Настройка объектов WinPlatform и AppEngine на архивирование данных

Чтобы настроить объекты WinPlatform и AppEngine на архивирование данных

1. Выполните захват объекта.
2. Запустите редактор объекта.
3. На странице **Инструментарий (Engine)** редактора объектов установите флажок **Разрешить хранение в архиваторе (Enable Storage to Historian)**.
4. В поле **Архиватор (Historian)** введите имя узла InSQL.
5. Укажите каталог в поле **Каталог промежуточного хранения в поле (History Store Forward Directory)**. При необходимости он будет создан при передаче объекта. Если оставить данное поле незаполненным, будет использован каталог по умолчанию.
6. Укажите в поле **Срок промежуточного хранения (Store Forward Deletion Threshold)**, спустя какой промежуток времени данные, записанные в каталог промежуточного хранения, должны быть удалены.
7. Сохраните сделанные изменения и закройте окно редактора.
8. Освободите объект.
9. Выполните пересылку объекта на компьютер использования, разрешив его сканирование.

Подробнее о каждом действии см. в главах "Объекты" и "Редакторы объектов". Подробнее о страницах редактора см. в справочной информации об объектах WinPlatform и AppEngine.

---

**Внимание!** Если пересылка объекта AppEngine будет выполнена до запуска приложения архиватора InSQL, данные в архиве не будут сохраняться до тех пор, пока не будет выполнена успешная регистрация в InSQL. При рассылке объектов AutomationObject, сканирование которых разрешено, значения некоторых атрибутов могут первоначально не сохраняться, до тех пор, пока они не будут зарегистрированы и переданы в InSQL.

---

## Настройка остальных объектов для архивирования данных

Каждый объект системы обладает некоторым набором атрибутов, значения которых могут сохраняться в архиве при их изменении. Вместе со значениями атрибутов сохраняются также сведения о качестве данных. Сохранённые в архиве записи могут впоследствии извлекаться для просмотра, анализа, построения трендов и других целей.

Чтобы значения атрибутов сохранялись в архиве, в параметрах соответствующего объекта AppEngine должна быть указана эта операция и, кроме того, он должен использоваться в Galaxy.

#### Чтобы объект пересылал данные для сохранения в архиве

1. Выполните захват объекта.
2. Запустите редактор объекта.
3. Для каждого атрибута, значения которого должны сохраняться в архиве, установите флажок **Архивировать (History)** и введите периодичность сохранения в поле **Периодичность сохранения (Force Storage Period)**.
4. Для атрибутов, не являющихся массивами (то есть для атрибутов Integer – целое, Float – вещественное или Double – двойной точности), дополнительно могут быть указаны значения полей **Допуск на значения (Value Deadband)**, **Максимум тренда (TrendHi)** и **Минимум тренда (TrendLo)**.
5. Сохраните сделанные изменения и закройте окно редактора.
6. Освободите объект в Galaxy.
7. Перешлите объект, разрешив для него сканирование, в хост-объект AppEngine, который также должен быть настроен на архивирование данных (см. предыдущий параграф).

Подробнее см. в Главах "Объекты" и "Редакторы объектов", подробнее о специализированных страницах редактора – в файле справки для данного типа объектов.

Значение **Допуск на значения (Value Deadband)** представляет собой указываемую в единицах измерения разницу между последним сохранённым и новым значением данных, при превышении которой новое значение будет сохранено в архиве (если разница между последним сохранённым и новым значением будет меньше этой величины, новое значение в архиве сохраняться не будет). Допуск, равный 0, допустим и является значением по умолчанию. "0" означает, что сохранённые в архиве данные должны измениться на некоторую величину.

---

**Внимание!** Изменение качества данных всегда приводит к созданию новой записи архива, независимо от того, изменилось ли значение данных или нет.

---

Значение, указываемое в поле **Периодичность сохранения (Forced Storage Period)**, представляет собой длительность интервалов времени в секундах, спустя которые значения атрибута сохраняются в архиве независимо от того, был ли превышен допуск на значения или нет. При ненулевой длительности выполняется регулярное сохранение значений атрибута. "0" в этом поле отменяет регулярное сохранение.

---

**Внимание!** Последовательные совпадающие пары "значение-качество" в действительности на диск не записываются.

---

Значение в поле **Максимум тренда (TrendHi)** определяет максимальное значение тренда, передаваемое клиентам.

Значение в поле **Минимум тренда (TrendLo)** определяет минимальное значение тренда, передаваемое клиентам.

Если во время работы приложения значение атрибута меняется, архивирование данных выполняется в следующих случаях:

- Когда атрибут числового типа (например double, float или integer) и новое значение отличается от последнего сохранённого на величину, превышающую допуск на значения, или если качество значения атрибута изменилось: из плохого (Bad), например, стало хорошим

(Good). Если предыдущего сохранённого значения нет, в архив заносится первоначальное значение (которое атрибут получил, например, после запуска системы).

- Когда атрибут является атрибутом типа качества (например enumeration – перечисление, string – строка или Boolean – логический), и значение или качество значения атрибута изменилось. Если предыдущего сохранённого значения не существует, в архив заносится первоначальное значение.
- Когда интервал времени, прошедшего с момента последнего сохранения, превышает длительность, указанную в поле **Периодичность сохранения (Force Storage Period)**. Если с момента запуска системы принудительного сохранения значений не было, оно выполняется немедленно.

---

**Примечание.** Механизм сравнения с допуском автоматически настраивается в соответствии с сохраняемым новым значением.

---

Новое значение атрибута вместе с отметкой времени и индикатором качества пересылаются архиватору InSQL, который записывает их на диск.

---

**Внимание!** При прекращении работы узла InSQL сохранение данных продолжается на локальный диск. После того как работоспособность узла InSQL будет восстановлена, локальные данные пересылаются ему с низким приоритетом. Если соединение объекта AppEngine с узлом InSQL нарушается, InSQL передаёт данные клиентам, указывая их плохое качество. При отмене использования объекта, для которого выполнялось архивирование значений атрибутов, последняя выборка данных записывается с индикатором плохого качества (Bad).

---

## Типы архивируемых атрибутов

Архивирование значений поддерживается для атрибутов следующих типов:

- Float (вещественный) – числовой.
- Double (вещественное двойной точности) – числовой: отображается в тип "float" архиватора InSQL. Если значение Double выходит за границы диапазона значений float, в архив записывается максимально допустимое значение float с индикатором качества данных "Uncertain" (неопределённое).
- Integer (целое) – числовой.
- Boolean (логическое) – не числовой.
- String (строковый) в кодировке Unicode – не числовой. Размер строки ограничен 512 символами. Более длинные строки усекаются и сохраняются с индикатором качества данных Uncertain" (неопределённое).
- CustomEnum (перечисление) – не числовой: отображается в тип "integer" архиватора InSQL.
- ElapsedTime (истёкшее время) – не числовой: отображается в тип "float" архиватора InSQL и преобразуется в секунды.

---

**Примечание.** Массивы и части массивов не поддерживаются.

---

Значения атрибутов перечисляемого типа архивируются как порядковые целые числа. Значения "NaN" атрибутов float и double преобразуются предварительно в значения NULL.

Все числовые величины пересылаются архиватору InSQL выраженными в единицах измерения, определённых для соответствующего атрибута. Архиватор InSQL не выполняет масштабирование полученных данных.

---



## Расширение набора архивируемых атрибутов

Указать дополнительные атрибуты для архивирования можно на странице **Расширения (Extensions)**, которая имеется в окне каждого редактора объектов. Подробнее см. главу "Расширение функциональных возможностей объектов".

## Изменение режима использования объекта и InSQL

Изменения в определении атрибутов AutomationObject при повторной его пересылке в место использования заставляют архиватор InSQL изменять параметры сохранения. Например, если единица измерения значений тэга вместо "Deg F" стала "Deg C", при повторной пересылке объекта это изменение будет записано в конфигурационной базе данных InSQL.

При отмене использования объекта, значения атрибутов которого записывались в архив, все накопленные к этому моменту сведения сохраняются и могут быть использованы в дальнейшем, даже если соответствующий объект AutomationObject больше использоваться в системе не будет.



## ГЛАВА 5

# Алармы и события

Пользователи IAS имеют возможность автоматизировать процессы обнаружения, уведомления, архивирования и просмотра как прикладных (технологических) событий и алармов, так и системных, то есть алармов и событий операционной системы и прикладных программ. События и алармы в контексте данного документа представляют собой любые изменения в состоянии рабочей системы, сведения о которых пересылаются в систему InTouch и заносятся в архив с помощью регистратора InTouch Alarm Logger.

В настоящей главе описываются:

- Различия между событиями и алармами.
- Способы конфигурирования поставщиков алармов.
- Методы генерации событий и алармов и уведомления клиентов.

## Содержание

- События и алармы
- InTouch как получатель сведений об алармах и событиях
- Конфигурирование объектов AutomationObject
- Объекты, способные генерировать алармы и события
- Различия между алармами и событиями в IAS и в InTouch

## События и алармы

Алармы и события являются разными понятиями, и ArchestrA "умеет" их различать.

Событие означает возникновение некоторых условий в определённый период времени. ArchestrA может обнаруживать события, сохранять о них сведения и передавать их различным клиентам.

Алармы представляют собой возникновение условий, рассматриваемых как аномальные (обычно неблагоприятные по своим последствиям), которые требуют немедленного вмешательства оператора. Алармы являются особым видом событий, которые характеризуются своим состоянием и должны подтверждаться пользователем. ArchestrA обеспечивает оперативное уведомление о возникновении алармов и предоставляет специальные средства просмотра сведений об алармах.

Примеры событий:

- Начало производственного процесса (например запуск производства новой серии).
- Изменение производственного параметра оператором (например задание новой уставки для термостата).

- Изменение технологом конфигурации рабочей системы (например применение нового объекта AutomationObject).
- Включение или отключение компонента системы (например объекта AppEngine).
- Регистрация пользователя в системе.
- Обнаружение отказавших программных компонентов (например неправильно функционирующего объекта ApplicationObject).

Примеры алармов:

- Превышение производственным параметром некоторого максимально допустимого значения (например максимально допустимой температуры).
- Переключение технологического устройства в нештатный режим работы (например самопроизвольная остановка рабочего насоса).
- Возникновение нештатного режима работы системных компонентов (например 99-процентная загрузка процессора в течение длительного периода времени).

Следующие ситуации не рассматриваются как события или алармы:

- Манипулирование конфигурационными параметрами в Репозитории Galaxy (например импортирование или захват объектов).
- Установка на компьютере приложения начальной загрузки Bootstrap.
- Пересылка события регистратору Alarm Logger. Иногда возникновение определённых системных событий может приводить к регистрации сообщений в системном журнале. Регистрация сообщений не рассматривается как событие.
- Открытие окна в InTouch.

Объекты ApplicationObject обладают механизмами генерации алармов и уведомления о возникновении событий. Для правильного выполнения этих функций параметры этих механизмов должны быть установлены в ИСП. Получатели алармов смогут осуществлять контроль над алармами, генерируемыми объектами AutomationObject, после активизации соответствующего объекта Area.

## Система InTouch как получатель сведений об алармах и событиях

Клиенты рабочей системы InTouch имеют возможность получать из Galaxy сведения об алармах. Для правильного функционирования подсистема алармов сервера IAS необходимо выполнение следующих условий:

- В узле клиента InTouch должна использоваться и функционировать WinPlatform, которая дополнительно должна быть сконфигурирована как поставщик алармов для InTouch.
- На базе этой WinPlatform должен функционировать один или несколько объектов Area.
- Оператор системы InTouch должен иметь возможность просматривать сведения об алармах, подтверждать алармы, а также разрешать и запрещать их генерацию из InTouch.
- Клиентский компонент InTouch встроен в окно и сконфигурирован как потребитель клиентов Galaxy.
- Клиент InTouch (WindowViewer) запущен и функционирует.

- Сканирование объекта-источника AutomationObject разрешено.
- Сканирование объекта Atea, в который входит объект-источник AutomationObject, разрешено.

---

**Примечание.** Сканирование объекта Atea не обязательно разрешать, если интересующий объект AutomationObject является объектом WinPlatform, AppEngine или DIObject. В этом случае достаточно разрешить сканирование только самого объекта-источника.

---

- Должна быть разрешена генерация алармов требуемым объектом AutomationObject.
- Аларм сервера IAS может быть сгенерирован объектом AutomationObject, входящим в объект Atea, или одним из объектов WinPlatform, AppEngine или DIObject. Потребитель алармов в InTouch должен подписаться на получение алармов от обоих наборов объектов при помощи распределённой системы алармов InTouch.
- Сервер IAS извещает об алармах распределённую систему алармов InTouch, которая, в свою очередь, извещает об этом приложение InTouch.
- Информация о новом неподтверждённом аларме, включая все требуемые поля, отображается в окне клиента InTouch. Новый аларм находится в состоянии "не подтверждён".
- Сведения о новом аларме записываются в архив событий InTouch.
- Параметры доступа в InTouch должны быть установлены в соответствии с параметрами доступа ArchestrA.
- Пользователь должен быть зарегистрирован в InTouch и иметь полномочия для подтверждения алармов генерирующего их объекта AutomationObject.
- Пользователь должен иметь возможность открыть окно алармов InTouch, выделить неподтверждённые алармы и подтвердить их, введя, при необходимости, свои примечания.
- Сервер IAS проверяет, имеет ли пользователь нужные полномочия для подтверждения аларма.
- Сведения о подтверждении аларма отображаются в других клиентах InTouch.
- Сведения о подтверждении аларма записываются в архив событий InTouch.
- После того как аларм будет подтвержден и возвратится в нормальное состояние, сведения о нём удаляются из окна алармов InTouch.

---

**Внимание!** Если полномочий для подтверждения аларма недостаточно, аларм, тем не менее, можно подтвердить, введя дополнительные примечания. Однако Galaxy в этом случае отклонит запрос на подтверждение аларма. В окне алармов InTouch он останется в состоянии "не подтверждён". Сведения о неподтверждении аларма будут записаны в архив событий InTouch, если пользователь, попытавшийся подтвердить его, является пользователем с допустимыми учётными данными в Galaxy. В противном случае отклонённое подтверждение не будет регистрироваться как событие.

---

## Запрет алармов объекта AutomationObject

Пользователь с достаточными полномочиями может запрещать и разрешать алармы объекта AutomationObject в окне InTouch. Для этого нужно

соответствующим образом установить значение атрибута AlarmModeCmd любого из следующих объектов:

- Объекта AutomationObject.
- Контейнера объекта AutomationObject (если он существует).
- Зоны объекта AutomationObject.

При этом в системе генерируется событие, обозначающее запрет генерации алармов, а сведения обо всех алармах данного объекта, находящихся в активном состоянии, удаляются из окна алармов InTouch.

При запрете алармов объекта AutomationObject запрещаются все его алармы. При запрете алармов в объекте Area запрещаются алармы всех вложенных объектов зоны и объектов AutomationObject, которые им принадлежат. При запрете алармов в контейнере объекта AutomationObject блокируются также алармы всех остальных вложенных в этом контейнере объектов AutomationObject.

## Разрешение алармов объекта AutomationObject

Для обработки алармов объекта AutomationObject нужно, чтобы атрибуты AlarmModeCmd и AlarmInhibit самого объекта, его контейнера и зоны имели разрешающее значение. При разрешении алармов генерируется соответствующее событие.

## Конфигурирование объектов AutomationObject

Генерация алармов входит в число функциональных возможностей шаблона объекта, однако они не реализуются до тех пор, пока не будут должным образом сконфигурированы в ИСР.

Конфигурирование объекта AutomationObject как поставщика алармов включает в себя выполнение следующих действий:

- Указания необходимости уведомления о каждом возникновении условий аларма (например о безуспешном выполнении выходной команды поворота задвижки вентиля).
- Изменения параметров объекта, в частности тех атрибутов, которые управляют выдачей уведомлений.
- Установки параметров алармов. Как правило, требуется определение значений таких полей, как Категория (Category), Приоритет (Priority) и Описание (Description).
- Задания предельных значений для определения аларма (например предельную задержку отклика обратной связи).

### Чтобы определить объект WinPlatform как поставщика алармов InTouch

1. Выполните захват объекта WinPlatform.
2. Запустите редактор объектов.
3. Установите флажок **Поставщик алармов InTouch (InTouch Alarm Provider)** на странице редактора **Общее (General)**.
4. Укажите в списке поля **Зоны алармов (Alarm Areas)** требуемую зону или оставьте поле незаполненным, чтобы означает выбор всех зон.

5. Сохраните сделанные изменения и закройте окно редактора.
6. Освободите объект, чтобы вернуть его в Galaxy.
7. Выполните пересылку объекта на целевой компьютер, разрешив для него сканирование.

## Алармы, определяемые пользователем

Пользователи ИСП могут дополнять объекты AutomationObject функциональными возможностями обнаружения алармов и выдачи уведомлений, первоначально в них отсутствовавшими. Это обеспечивается с помощью расширений скриптов и алармов. Подробнее см. главу "Расширение функциональных возможностей объектов".

## Объекты, способные генерировать алармы и события

После того как сканирование экземпляров объектов AutomationObject на целевом компьютере разрешено, начинается проверка условий возникновения определённых для этих объектов алармов и событий. При возникновении какого-либо аларма или события сведения о вызвавших его условиях передаются распределителям алармов и событий, которыми являются другие объекты AutomationObject, функционирующие в том же самом объекте AppEngine.

В число таких объектов входят:

- Объекты Atea: все объекты домена и зоны передают сведения об алармах объекту Atea, который распределяет её потребителям алармов и событий.
- Объекты WinPlatform: передают сведения о собственных алармах и событиях.
- Объекты AppEngine: передают сведения о собственных алармах и событиях.
- Объекты DIObject: передают сведения о собственных алармах и событиях.

Объекты WinPlatform, AppEngine и DIObject не передают сведения объекту Atea, даже если они входят в него. Это позволяет клиентам получать уведомления об алармах независимо от состояния объектов Atea. В частности, объект WinPlatform способен передавать данные об алармах, даже если на компьютере не функционируют объекты Atea.

Распределители алармов и событий создают списки всех активных и неактивных, но ещё не подтверждённых алармов. Списки событий ими не ведутся, а информация передаются соответствующим клиентам сразу же в момент возникновения события.

## Объекты AutomationObject типа Area

Объекты AutomationObject типа Area имеют большое значение в распределении сведений о событиях и алармах. Все объекты AutomationObject принадлежат какому-либо объекту Area. В объекте Area могут использоваться вложенные объекты данного типа. Все клиенты-потребители алармов и событий должны устанавливать соединение с каким-либо набором объектов Area. Таким образом, они играют ключевую роль в группировании сведений об алармах и событиях и распределении их среди клиентов. С их помощью осуществляется контроль состояния соответствующих зон.

## Подписка на получение сведения об алармах и событиях

Чтобы использовать сведения об алармах и событиях зоны, клиенты должны подписаться на получение этих сведений из данной зоны. Подписка на получение сведений из объекта Atea обычно означает подписку на получение сведений от всех объектов-распределителей внутри этого объекта. При наличии вложенных объектов Atea от них также передаются сведения об алармах, и если в объект Atea входят объекты WinPlatform, AppEngine или DIObject, клиент будет получать сведения об алармах и от них.

Получив сигнал клиента о подписке, распределитель уведомлений предоставляет клиенту следующую информацию:

- Перечень всех существующих состояний аларма, включая неподтвержденные возвраты в нормальное состояние.
- Сведения об изменениях состояний аларма. В число подобных изменений входят переход и выход из состояния аларма (возврат в нормальное состояние), а также изменение флажка подтверждения.
- Сведения о возникновении события.

При выдаче запроса на подписку фильтры не учитываются (например, нельзя подписаться на получение алармов указанного приоритета). После подписки клиенту будет передаваться вся получаемая распределителем информация. Фильтры должны организовываться в приложении-клиенте.

## Разрешение и блокирование алармов

Обнаружение алармов (но не событий) в рабочем приложении можно разрешать и запрещать. Это действие может выполняться на уровне зоны, контейнера и собственно объекта.

---

**Примечание.** Выборочное блокирование и разрешение алармов одного объекта не поддерживается.

---

В рабочей системе возможны следующие режимы обнаружения алармов:

- Enabled (разрешение алармов): сведения об алармах объекта передаются клиентам и записываются в архив обычным образом.
- Disabled (блокирование алармов): обнаружение алармов объекта не выполняется. По существу, в данном случае осуществляется возврат в нормальное состояние (до тех пор, пока обнаружение алармов не будет вновь разрешено).

---

**Внимание!** Режим Silenced (подавление алармов) зарезервирован для будущего использования. В текущей версии он аналогичен режиму блокирования алармов.

---

В следующей таблице приведены сведения о конечном состоянии обнаружения алармов с учётом состояния других объектов в иерархии.

Режим обнаружения алармов зоны	Режим обнаружения алармов контейнера	Указанный режим обнаружения алармов объекта	Конечный режим обнаружения алармов объекта
Enabled	Enabled	Enabled	Enabled
Disabled	Любой	Любой	Disabled
Любой	Disabled	Любой	Disabled
Любой	Любой	Disabled	Disabled



## Функционирование поставщика алармов при отключении сети

Сконфигурированный как поставщик алармов InTouch объект WinPlatform пересылает клиентам сведения об алармах через распределённую систему алармов InTouch, когда соединение с объектом Asea, с которым он взаимодействует, нарушается. Как правило, такая ситуация возникает при отсутствии сетевых соединений с компьютерами, на которых функционируют соответствующие объекты Asea.

При отказе сети объект WinPlatform как поставщик алармов InTouch генерирует алармы для каждой из отключившихся зон Asea (включая всю иерархию распределения сведений об алармах). Каждый из них представляет собой высокоприоритетный аларм, в данных о котором указано название зоны, обмен информацией с которой прекратился. Все такие алармы, предупреждающие о проблеме каналов связи, должны быть подтверждены.

Несмотря на продолжение записи в архив, все сведения о текущих алармах отключившейся зоны из сводного списка клиента InTouch удаляются. Подтвердить эти алармы невозможно. После восстановления соединений с зонами клиентам будут переданы все сведения о неподтверждённых алармах, сгенерированных в этих зонах.

## Различия между алармами и событиями в IAS и в InTouch

События и алармы имеются как в InTouch, так и в сервере IAS. Их сходство и различие описаны в следующей таблице.

Характеристика	Система InTouch	Сервер IAS
Определение и обнаружение аларма	Внутри тэга	Внутри объекта AutomationObject
Классы алармов (клиентское поле)	Поддерживаются и обнаруживаются только определённые классы алармов: DSC, VALUE, DEV, ROC, SPC	Системного разграничения классов нет. Алармы связаны с логическими величинами, состояние которых может быть изменено любыми средствами. Представляют собой отображение категорий.
Типы алармов (подклассы) (клиентское поле)	Discrete, LoLo, Lo, Hi, HiHi, MinorDev, MajorDev, ROC, SPC. Клиентское поле.	Подклассов нет (ближайшее понятие – название примитива аларма ".PVNAlarm"). Представляют собой отображение категорий.
Приоритет (клиентское поле)	1-999 (самый важный – 1).	0-999 (самый важный – 0). Приоритет уровня 0 соответствует приоритету уровня 1 в InTouch.
Название (клиентское поле)	Название аларма совпадает с названием тэга	Объект.атрибут
Примечание (клиентское поле)	Примечание в описании тэга = короткое описание	Краткое описание в объекте AutomationObject

		или сообщение об аларме.
Группа	Группы алармов представляют собой клиентское средство фильтрации. Подгруппы должны находиться в том же узле InTouch.	Группирование алармов отсутствует. Объект Area представляет собой некоторое подобие группирования алармов. Вложенные объекты Area могут находиться на разных узлах.
Состояние	UNACK, ACK, RTN	Аналогичные состояния
Значение, проверочное значение	Вместе с сообщениями об алармах передаются только неизменяемые величины.	Возможность передачи как неизменяемых значений, так и динамических ссылок.
Подтверждение	Клиентам передаются сведения обо всех алармах, которые должны подтверждаться независимо от их приоритета.	Клиентам передаются сведения обо всех алармах, которые должны подтверждаться независимо от их приоритета.
Запись в архив	Сведения об изменениях состояния аларма записываются в архив событий и отображаются в клиенте архивирования.	Сведения об изменениях состояния аларма записываются в архив событий и отображаются в клиенте архивирования.

# Расширение функциональных возможностей объектов

Каждый импортированный в Galaxy объект обладает определённым набором функциональных возможностей, предусмотренных разработчиком объекта. Вместе с тем существует возможность расширения этих возможностей пользователем – на странице расширений объекта, которая присутствует в окне каждого редактора объектов.

На этой странице могут быть определены новые функциональные возможности объекта (без изменения первоначального набора). Новые функции называются расширениями объекта и могут добавляться в производные шаблоны и экземпляры объектов. Базовые шаблоны расширить нельзя.

Существует три типа расширений объектов:

- Скрипты, которые определяются на странице **Скрипты (Scripts)** окна редактора объектов.
- Пользовательские атрибуты, которые определяются на странице **Пользовательские атрибуты (UDAs)** окна редактора объектов.
- Расширения атрибутов, которые используются для расширения функциональных возможностей уже существующих атрибутов объекта и определяются на странице **Расширения (Extensions)** окна редактора объектов.

В настоящей главе описываются методы определения скриптов, пользовательских атрибутов и расширений атрибутов объектов для расширений функциональных возможностей объектов в приложениях IAS.

Подробнее см. Руководство по разработке проектов FactorySuite A<sup>2</sup>.

## Содержание

- Страницы расширений в окне редактора объектов
- Наследование расширений
- Скрипты
- Определяемые пользователем атрибуты
- Расширения атрибутов
- Язык скриптов QuickScript .NET

## Страницы расширений в окне редактора объектов

Расширение функциональных возможностей объектов выполняется на специальных страницах, которые присутствуют в окне любого редактора

конфигурации, с закладками **Скрипты (Scripts)**, **Пользовательские атрибуты (UDAs)** и **Расширения (Extensions)**.

Более подробно эти страницы описаны в следующих параграфах.

## Наследование расширений

Расширение функциональных возможностей поддерживается только для производных шаблонов и экземпляров объектов. Базовые шаблоны изменить нельзя. При расширении функциональных возможностей с порождёнными объектами выполняются следующие действия:

- Если в шаблоне, по которому уже созданы объекты, название расширения изменяется, оно будет изменено таким же образом и во всех порождённых объектах. Указанное изменение будет произведено в момент освобождения шаблона.
- При удалении из шаблона расширения оно также будет удалено и из всех созданных на основе этого шаблона объектов. Указанное действие будет произведено в момент освобождения шаблона.
- Добавление нового расширения в производный шаблон приведёт к добавлению этого расширения во все созданные на его основе объекты.
- Добавление в производные объекты расширений, дублирующих расширения родительских объектов (с тем же именем и того же типа) не допускается.
- Добавление расширений с дублирующими именами не допускается.
- Освобождение шаблона с новым расширением, имя которого конфликтует с именем расширения одного из атрибута в порождённом объекте, допустимо. Определение расширения в шаблоне отменяет определение расширения в порождённом объекте.

## Скрипты

Скрипты представляют собой группы команд и логических операций, выполнение которых осуществляется при выполнении некоторых условий (например при нажатии клавиши, при открытии окна или при изменении значения атрибута и т.д.). С помощью скриптов возможна реализация самых разнообразных специализированных системных функций.

Указанные в скрипте команды и операции будут выполняться, если содержащий их объект используется в приложении и:

- сканирование объекта во время работы приложения разрешено или
- статус сканирования или запуска/останова изменяется.

Обычно скрипт исполняется при изменении значений тех или иных атрибутов объекта, но вместе с тем он может запускаться другими скриптами при изменении значений атрибутов более чем одного объекта.

---

**Примечание.** Значения атрибутов объектов, находящихся вне объекта AppEngine, при первом исполнении скрипта "On Startup" (При запуске), неизвестны. Как правило, они становятся известными только спустя несколько периодов сканирования. Поэтому перед выполнением каких-либо важных операций следует включить проверку качества используемых данных для проверки их доступности.

---

Поддержка общих переменных, которые используются аналогично атрибутам расширенного объекта, в скриптах отсутствует. Сложные функции могут быть реализованы на основе скриптов и атрибутов, определяемых пользователем. Подробнее см. параграф "Определяемые пользователем атрибуты".

Назначение страницы **Скрипты (Scripts)** означает определение скриптов, которые будут исполняться в рабочей системе при возникновении тех или иных условий.

Скрипты могут быть определены для любого производного шаблона или экземпляра объекта Galaxy.

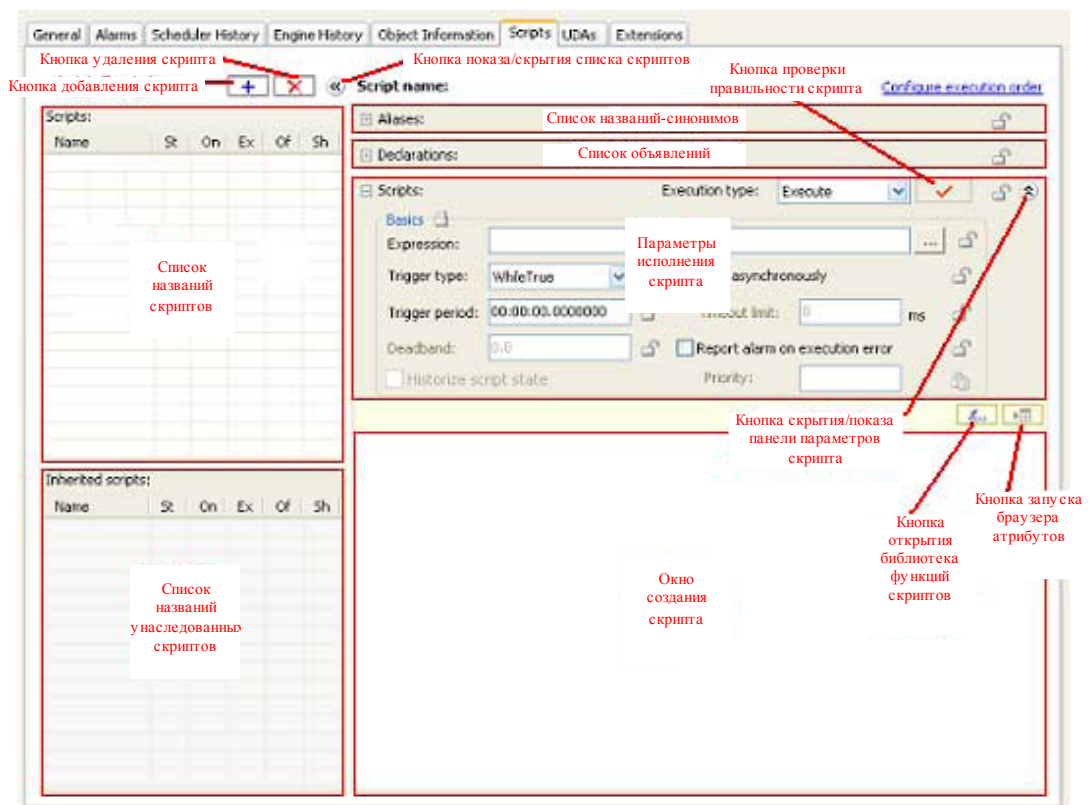
Скрипты сервера промышленных приложений IAS обеспечивают поддержку следующих функциональных возможностей системы объектов AutomationObject:

- Выполнение алгоритмов диспетчерского управления.
- Выполнения алгоритмов серийного производства.
- Определение сложных алармов.
- Обмен информацией с базой данных.
- Генерация и издание отчётов.
- Поддержка пространства имён .NET.

Подробнее о функциях скриптов и их аргументах, а также примеры их использования см. параграф "Язык скриптов QuickScript .NET".

## Страница Скрипты (Scripts)

На странице **Скрипты (Scripts)** расположено шесть функциональных панелей и набор кнопок, описание которых приведено ниже.



Основными функциональными компонентами страницы скриптов являются:

- **Имена скриптов (Scripts Name):** перечень всех скриптов, определённых в настоящий момент для данного объекта. В столбцах таблицы отмечен способ запуска скриптов: Startup (при запуске), On Scan (при включении сканирования), Execute (при исполнении), Off Scan (при выключении сканирования), Shutdown (при останове).
- **Названия унаследованных скриптов (Inherited Scripts Name):** перечень всех скриптов, определённых для родительского объекта. Эти скрипты связываются с текущим объектом автоматически. В столбцах таблицы отмечен способ запуска скриптов: Startup (при запуске), On Scan (при включении сканирования), Execute (при исполнении), Off Scan (при выключении сканирования), Shutdown (при останове).
- **Псевдонимы (Aliases):** перечень названий синонимов для текущего скрипта. Имена-синонимы представляют собой более короткие описательные обозначения ссылок ArchestrA, которые могут использоваться внутри скрипта.
- **Объявления (Declarations):** панель, в которую могут добавляться операторы определения переменных (с типом "DIM MyArray[1] as Float;"). Эти объявления сохраняются и действительны только для того скрипта, в котором они указаны.
- **Параметры исполнения:** панель определения условий запуска скрипта в рабочем приложении. Типы скриптов и способы запуска описаны в процедуре создания скриптов и их связывания с объектом.
- **Создание скрипта:** панель редактирования текста скрипта.
- **Определение порядка исполнения (Configure Execution Order):** при нажатии этой кнопки появляется окно, в котором определяется порядок исполнения скриптов объекта (унаследованных и собственных).

---

**Внимание!** Порядок выполнения скриптов фиксируется на каждом этапе разработки шаблона, производного шаблона и объекта. В частности, набор скриптов, определённых для шаблона, интерпретируется в панели параметров исполнения скрипта производного шаблона как единый упорядоченный блок. Скрипты производного шаблона могут исполняться как до, так и после этого блока. Аналогично, все скрипты этого шаблона будут интерпретироваться в следующем производном шаблоне или экземпляре объекта также как единый блок скриптов.

---

- **Сохранять состояние скрипта в архиве (Historize Script State):** установите этот флажок, если информация о состоянии скрипта должна пересылаться архиватору ArchestrA InSQL.

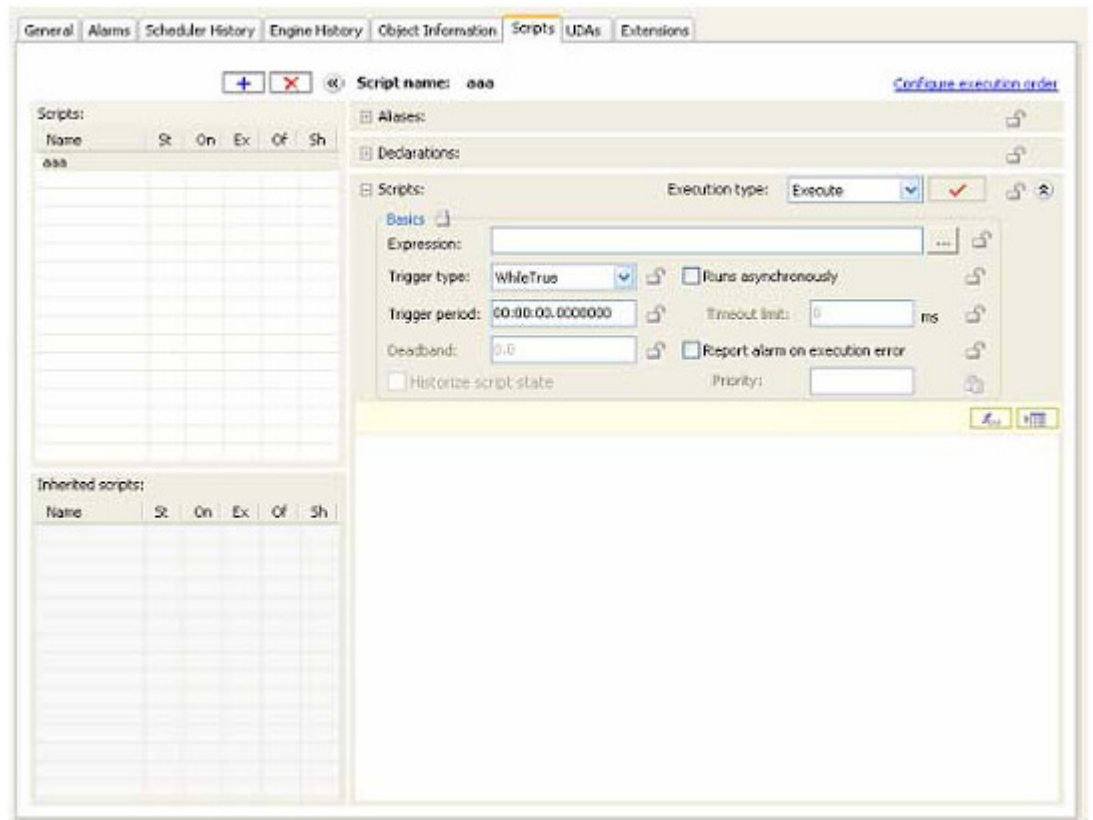
В число функциональных кнопок входят следующие кнопки:

- Кнопка добавления скрипта: при нажатии этой кнопки осуществляется определение нового скрипта объекта.
- Кнопка удаления скрипта: при нажатии этой кнопки осуществляется удаление выделенных в списке скриптов.
- Кнопка показа и скрытия списков скриптов: при последовательном нажатии этой кнопки производится показ и скрытие списка собственных и унаследованных скриптов объекта.
- Кнопка проверки правильности скрипта: при нажатии этой кнопки производится проверка синтаксиса введённых операторов скрипта.
- Кнопка показа и скрытия панели параметров исполнения: при последовательном нажатии этой кнопки производится показ и скрытие панели параметров исполнения скрипта.

- Кнопка открытия окна библиотеки скриптовых функций: при нажатии этой кнопки появляется окно браузера скриптовых функций для выбора требуемой функции.
- Кнопка запуска браузера атрибутов: при нажатии этой кнопки появляется окно браузера атрибутов, в котором пользователь может выбрать нужный атрибут и свойство объекта для использования в скрипте.

### Чтобы создать скрипт и связать его с объектом

1. Щёлкните символ "+" на странице **Скрипты (Scripts)** редактора объектов. В списке названий скриптов появится новый элемент. Удалить скрипт из списка можно, выделив его название и нажав кнопку с символом "x" красного цвета (сразу несколько скриптов можно выделить, если щёлкать их при нажатых клавишах **Shift** или **Ctrl**).
2. Введите название нового скрипта и нажмите клавишу **Enter (Ввод)**. Страница скриптов примет следующий вид (в предположении, что был добавлен новый скрипт с названием "aaa").



3. Укажите способ запуска скрипта в рабочем приложении. Возможны следующие типы: Startup (при запуске), On Scan (при включении сканирования), Execute (при исполнении), Off Scan (при выключении сканирования), Shutdown (при останове). Выбор любого из них, за исключением "Execute" приведёт к тому, что поля панели **Условия (Basics)** станут недоступными. Таким образом, данный скрипт будет запускаться всегда при запуске объекта, в начале сканирования, при окончании сканирования или при останове функционирования объекта. При выборе "Execute" поля панели условий запуска становятся доступными.

**Примечание.** Скрипты не могут запускаться чаще, чем один раз на протяжении интервала сканирования объекта AppEngine, для которого они определены или который содержит вложенный объект, с которым эти скрипты связаны.

4. Для скриптов со способом запуска "Execute" укажите вид скрипта в поле **Вид запуска (Trigger Type)**. В зависимости от заданного вида может потребоваться определить значение поля **Выражение (Expression)** и/или полей **Периодичность запуска (Trigger Period)** и **Мёртвая зона (Deadband)**. Скрипт будет запускаться при выполнении условий, определяемых значениями этих полей. Подробнее см. следующую таблицу.

**Примечание.** Периодичность запуска нужно указывать в следующем формате: дни:часы:секунды.доли\_секунд. Например, периодичность запуска, равная трём дням, пяти часам и десяти с половиной секундам, может быть задана как "03:05:10.5000000".

Вид запуска	Описание
Periodic (Периодически)	Скрипт выполняется по прошествии интервалов времени, определённых значением поля <b>Периодичность запуска (Trigger Period)</b> . При нулевом (0) значении поля скрипт запускается в каждом интервале сканирования. Логическое выражение для скриптов данного типа не требуется.
While True (Пока истинно)	После того как сканирование объекта будет разрешено, значение логического выражения вычисляется в следующем интервале сканирования объекта AppEngine. Скрипт запускается, если это значение равно "True", после чего периодически выполняется через указанный интервал времени, пока данное значение логического выражения сохраняется. Указание периодичности запуска обязательно; при нулевом значении значение выражения определяется в начале сканирования объекта AppEngine, при ненулевом – через указанный интервал времени.
On True (Переход в "истину")	После того как сканирование объекта разрешено, в следующем интервале сканирования начинается вычисление значения логического выражения. Скрипт запускается в момент его изменения с "False" на "True".
On False (Переход в "ложь")	После того как сканирование объекта разрешено, в следующем интервале сканирования начинается вычисление значения логического выражения. Скрипт запускается в момент его перехода из "True" в "False".
Data Change (При изменении данных)	Скрипт будет исполняться при изменении значения или качества (достоверности) выражения. Значение выражения вычисляется как число одного из следующих типов: integer (целое), real (вещественное), time (время), elapsedtime (прошедшее время), string (строка символов), double (число с двойной точностью), Boolean (логическое), custom enumeration (перечислитель) и quality(качество). Для выражений всех типов может быть указан допуск, при этом он выражается как число с двойной точностью. Единицей измерения значений "time" и "elapsedtime" является миллисекунда. Для символьных выражений допуски игнорируются, поскольку любые изменения (даже регистра символов: "ABC" → "abc") всегда интерпретируются как значительные. Учитываются также лишь существенные изменения качества (например переход качества "Good" или "Uncertain"

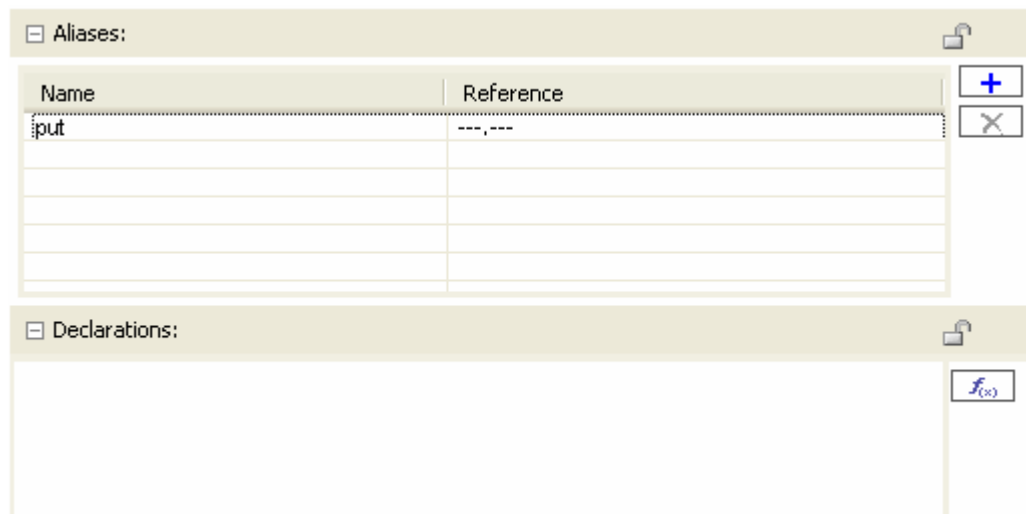


	в качество "Bad" или "Initializing" и наоборот). После разрешения сканирования объекта запускаемые изменениями данных скрипты исполняются в следующем интервале сканирования объекта AppEngine, а также в последующих интервалах при изменении значения или качества данных.
While False (Пока ложно)	После того как сканирование объекта разрешено, значение логического выражения вычисляется в следующем интервале сканирования объекта AppEngine. Скрипт запускается, если это значение равно "False", после чего периодически выполняется через указанный интервал времени, пока данное значение логического выражения сохраняется. Указание периодичности запуска обязательно; при нулевом значении значение выражения определяется в начале сканирования объекта AppEngine, при ненулевом – через указанный интервал времени.

**Примечание.** Об употреблении пользовательских атрибутов см. параграф "Определяемые пользователем атрибуты".

**Примечание.** Выражения должны представлять собой только один логический оператор, в котором может осуществляться только один вызов синхронных функций. Рекомендуется избегать использования в скриптах функций, обладающих побочными эффектами, поскольку это может привести к непредсказуемым результатам.

1. При необходимости установите флажок **Исполняется асинхронно (Runs Asynchronously)** и значение поля **Предел таймаута (Timeout Limit)**. (Подробнее см. параграф "Синхронное и асинхронное исполнение скриптов".) Установите флажок **Генерировать аларм при ошибке исполнения (Report Alarm on Execution Error)** и укажите в поле **Приоритет (Priority)** нужный приоритет, для того чтобы система предупреждала пользователя о возникновении ошибок исполнения скриптов. Установите флажок **Сохранять состояние скрипта в архиве (Historize Script State)**, если сведения о состоянии скрипта должно сохраняться в архиве приложения.
2. Введите в панель **Объявления (Declarations)** объявления переменных, которые будут использованы в теле скрипта.



3. Определите в списке поля **Псевдонимы (Aliases)** имена-синонимы. Применение имён-синонимов упрощает текст скрипта и обеспечивает

его блокирование на уровне шаблона. Впоследствии при создании экземпляра объекта достаточно будет определить внешние атрибуты для этих имён. Чтобы добавить новый элемент в список, щёлкните символ "+". Имя нового элемента будет оказано в режиме изменения. Двойной щелчок записи в столбце **Ссылка (Reference)** также переводит её в режим изменения. Введите для отмеченного имени-синонима соответствующую ссылку. Можно также нажать кнопку в конце поля Reference и выбрать нужный атрибут объекта в окне браузера атрибутов. Чтобы удалить элемент из списка, выделите его и щёлкните символ "x" красного цвета. Можно выделить сразу несколько элементов, если при их выделении держать нажатой клавишу **Shift** или **Ctrl**.

4. Введите в окно создания скриптов текст скрипта. Для вставки в тело скрипта ссылок на функции и атрибуты объектов можно использовать кнопки открытия окна библиотеки скриптовых функций и запуска браузера атрибутов.
5. Для проверки синтаксиса используемых операторов нажмите соответствующую кнопку.
6. Чтобы определить порядок выполнения скриптов, определённых для данного объекта, щёлкните **Определение порядка исполнения (Configure Execution Order)**. Для скриптов, выполняющихся асинхронно, эта функция недоступна. Если новый скрипт добавляется к объекту, который был создан по шаблону с уже определёнными скриптами, он по умолчанию будет исполняться после них.
7. После создания скрипта и определения порядка его исполнения сохраните введённую информацию и закройте окно редактора объектов. После пересылки объекта в место использования и разрешения его сканирования скрипты будут исполняться, как запланировано.

Создание скриптов имеет следующие особенности:

- Имя скрипта в окне редактора может быть изменено в любой момент.
- Ограничений на длину текста скрипта нет.
- При выборе функции в окне библиотеки её вызов вставляется в тело скрипта вместе с соответствующими аргументами.
- Сохранение скрипта с синтаксическими ошибками возможно. Тем не менее, переслать объект в место использования можно только после устранения всех ошибок.
- Выполняется также проверка семантики, позволяющая избежать употребления синтаксически правильных, но семантически неверных конструкций, таких как двойное объявление одной и той же переменной (в одном и том же блоке переменная может объявляться только один раз).
- Если при исполнении скрипта в рабочем приложении произойдёт какая-либо ошибка, исполнение скрипта будет прекращено и возобновлено в следующем периоде сканирования объекта AppEngine.

## Синхронное и асинхронное исполнение скриптов

По умолчанию, в рабочем приложении скрипты исполняются в синхронном режиме, то есть последовательно (если сканирование объекта разрешено).

В асинхронном режиме несколько скриптов исполняются одной и той же нитью с низким приоритетом. Эти скрипты могут быть только "Execute" и могут запускаться независимо друг от друга. Максимальное количество независимых нитей устанавливается с помощью редактора конфигурирования объекта AppEngine.

---

**Примечание.** В следующем периоде сканирования объекта AppEngine возникает таймаут исполнения асинхронных скриптов, даже если таймаут, указанный для скрипта, по длительности меньше, чем период сканирования этого объекта.

---

## Типы ошибок исполнения

Ошибками исполнения называются любые ошибки, которые могут быть обнаружены блоком исполнения скриптов и которые приводят к их аварийному завершению.

---

**Примечание.** Алармы генерируются при обнаружении ошибок только в скриптах "Execute".

---

Блок выполнения скриптов определяет возникновение следующих ошибок:

- **Timeout (таймаут):** для предотвращения непрерывного исполнения скрипта в течение всего цикла сканирования объекта AppEngine (то есть для предотвращения бесконечных циклов). Таймаут устанавливается в редакторе объекта для всех синхронных скриптов, определяемых как для самого объекта AppEngine, так и для вложенных в него объектов. Таймауты для асинхронных скриптов устанавливаются на странице **Скрипты (Scripts)** окна редактора объекта. При том его длительность будет определяться длительностью основного объекта AppEngine.
- **Overflow (переполнение):** этот тип применим только для данных целого типа, вещественного и с двойной точностью.
- **Division by zero (деление на ноль):** этот тип применим только для данных целого типа.
- **.NET call execution (вызов функции .NET):** сведения об ошибке, возникающей во время исполнения функции .NET, передаются регистратору Logger. Кроме того, генерируется аларм "Boolean condition" (логическое условие).

## Вызов функций в скриптах

Архитектура IAS поддерживает применение в скриптах вызовов заранее скомпилированных функций. Предпочтительными языками написания скриптовых функций являются языки архитектуры .NET (например, VB.NET или C#), хотя их можно создавать и с помощью других языков, после чего импортировать как WDF-файлы (унаследованные функции InTouch) или COM-объекты (C++ и VB).

Функции, которые можно использовать в скриптах, хранятся в виде библиотек. Они сгруппированы по назначению и могут вызываться как из самих скриптов, так и из других функций. Состояние функций между вызовами не сохраняется (исключение: если объявлена переменная, состояние секции будет сохраняться между обращениями к функции), они могут иметь как входные, так и выходные аргументы и при необходимости возвращать значения.

По умолчанию, для разработки скриптов доступны следующие шесть бинарных библиотек:

- Библиотека математических функций.
- Библиотека функций обработки строк символов.
- Библиотека системных функций.
- Библиотека функций различного назначения.
- Библиотека типов.

- Библиотека функций WWDDE.

Чтобы вставить в текст скрипта вызов той или иной функции, установите курсор в нужное место и щёлкните кнопку открытия библиотеки функций



, выберите в появившемся окне нужную функцию и щёлкните **ОК**.

Если вставленный текст должен быть уточнён (доопределён) пользователем, он будет выделен.

**Внимание!** Новые библиотеки функций могут быть созданы с помощью Visual Studio. Возможно также импортирование библиотек COM-объектов .NET, созданных с помощью Visual Studio версии 6 или более ранней, но в отличие от компонентов .NET, автоматическая пересылка библиотек COM-объектов вместе с соответствующим объектом не выполняется. Эти объекты нужно устанавливать и регистрировать в операционной системе вручную, перейдя в соответствующий каталог и введя DOS-команду "regsvr32 <название библиотеки .DLL>". Можно также экспортировать их в виде файла с расширением .aaSLIB (cab-файл), изменить XML-файл в этом cab-файле, определив библиотеку как COM-объект, требующий регистрации, и затем импортировать файл с расширением .aaSLIB повторно. Нужные компоненты будут использованы и зарегистрированы. Процедура импортирования, экспортирования и редактирования файла с расширением .aaSLIB описана далее.

**Примечание.** В языке QuickScript .NET аргументы функции должны указываться в скобках. Подробнее см. параграф "Язык скриптов QuickScript .NET".

## Рекомендации по разработке и использованию библиотек скриптовых функций

При разработке и использовании библиотек скриптовых функций рекомендуется учитывать следующие условия:

- Вложенные общие структуры и классы в импортированных библиотеках скриптовых функций не поддерживаются.
- Названия библиотек функций должны удовлетворять соответствующим требованиям (см. параграф "Допустимые имена и символы"). Во избежание конфликтов наименований библиотек различных поставщиков перед названием библиотеки нужно указывать имя поставщика (в формате "Поставщик.Библиотека.dll).
- Принципы разработки бинарных библиотек аналогичны принципам разработки функций InTouch на языке Си с помощью инструментальных средств InTouch Extensibility Toolkit.
- Скрипты исполняются внутри того же процесса, что и программное обеспечение объекта, с которым они связаны. Это значит, что скрипты с ошибками могут нарушать функционирование объекта AppEngine.
- Разработка библиотек скриптов должна включать следующие этапы:
  1. Разработка библиотеки в инструментальной среде, такой как Visual Studio .NET.
  2. Импортирование библиотеки в ИСР.
  3. Проверка использования функции в скрипте тестового объекта.
  4. Пересылка тестового объекта в объект AppEngine, не задействованный в производственной системе.
  5. Отладка и тестирование скрипта.
  6. Доработка, если нужно, скрипта.

7. Отмена использования тестового объекта со скриптом.
8. Прекращение функционирования тестового объекта AppEngine, для того чтобы структура .NET была выгружена из памяти.

---

**Внимание!** Выполнение шага 8 обязательно, так как в противном случае никакие изменения в библиотеке скриптовых функций не будут иметь силу.

---

9. Повторный запуск тестового объекта AppEngine.
10. Продолжение работы, начиная с шага 1.

#### **Чтобы импортировать библиотеку скриптовых функций**

1. Поместите курсор мыши на **Импортирование (Import)** меню Galaxu и выполните команду **Script Function Library (Библиотека скриптовых функций)** открывшегося меню.
2. В окне **Import Script Function Library (Импортирование библиотеки скриптовых функций)** найдите нужный файл библиотеки (с расширением .aaSLIB, .dll, .wdf, .tlb или .olb), выделите его и щёлкните **Открыть (Open)**, чтобы импортировать библиотеку, или кнопку **Отмена (Cancel)**, чтобы отменить операцию.
3. Щёлкните **ОК** в появившемся окне сообщения. Указанная библиотека будет импортирована в Galaxu, после чего содержащиеся в ней функции станут доступными во всех ИСП, подключенных к Galaxu, то есть при открытии окна библиотеки на странице **Скрипты (Scripts)** окна редактора объектов.

Если имя импортируемой библиотеки совпадает с именем уже существующей в Galaxu, появится окно с запросом замены существующей библиотеки или переименования импортируемой. При замене существующей библиотеке и повторной рассылке объектов, которые вызывают содержащиеся в ней функции, нужно повторно запустить объект AppEngine, для того чтобы новая библиотека стала действительной.

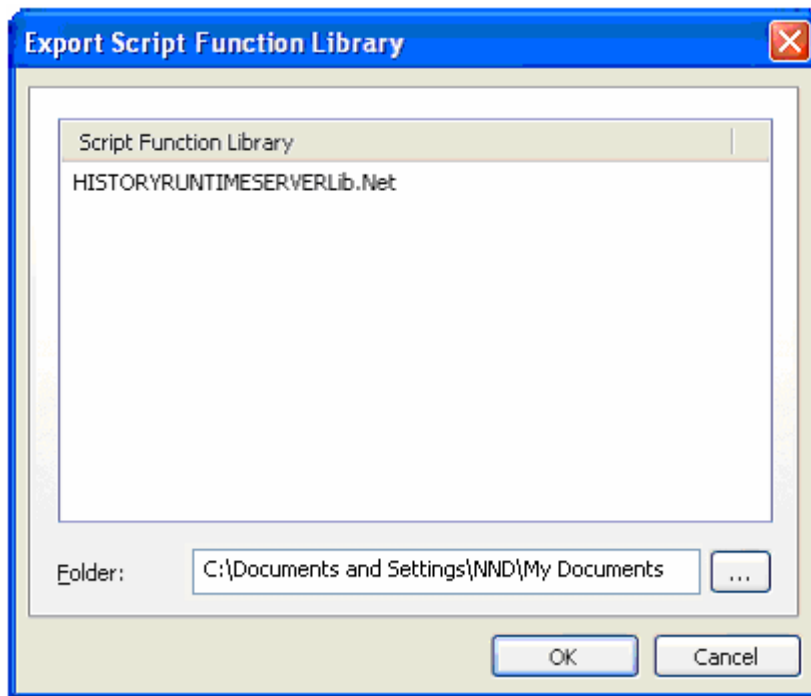
---

**Внимание!** При экспортировании объектов AutomationObject соответствующие функции не экспортируются, если они входят в состав импортированной библиотеки. Чтобы импортировать такие объекты в другую Galaxu, нужно сперва вручную экспортировать нужные библиотеки функций из исходной Galaxu, а затем импортировать их в систему назначения. Только после этого скрипты, связанные с импортированными объектами, будут исполняться правильно.

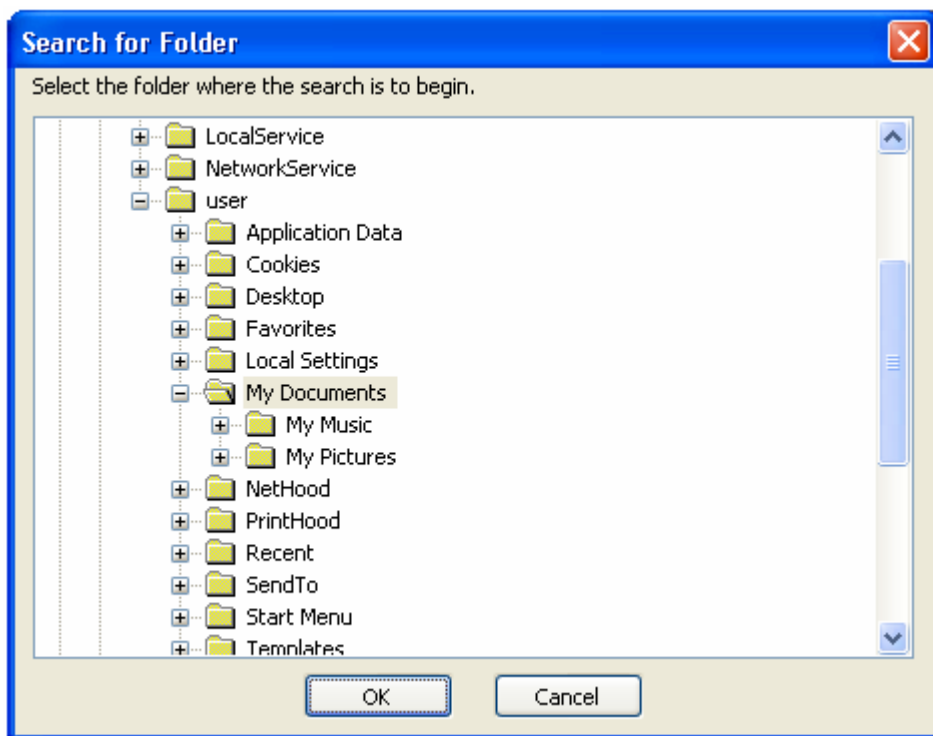
---

#### **Чтобы экспортировать библиотеку скриптовых функций**

1. Поместите курсор мыши на **Экспортирование (Export)** меню Galaxu и выполните команду **Script Function Library (Библиотека скриптовых функций)** открывшегося меню. Появится окно **Export Script Function Library (Экспортирование библиотеки скриптовых функций)**.



2. Выберите из списка нужную библиотеку. Если её имя в списке не отображается, введите название папки или щёлкните, для её поиска, кнопку с многоточием. Появится окно **Поиск папок (Search for Folder)**.



3. Найдите нужную папку и щёлкните **ОК**, сначала в окне поиска, а затем – в окне экспортирования библиотек. Появится стандартное окно операционной системы **Сохранить как (Save As)**.
4. Укажите требуемый каталог и имя файла экспортирования библиотеки и щёлкните **Сохранить (Save)**. Библиотека скриптовых функций будет сохранена в файле с расширением **.aaSLIB**.

**Чтобы вручную изменить XML-файл в .aaSLIB-файле с целью обозначения библиотеки скриптовых функций как COM-объекта, требующего регистрации**

1. В окне Проводника Windows переименуйте файл ??? .aaSLIB в файл ??? .aaSLIB.cab, где символы "???" представляют собой имя модифицируемой библиотеки.
2. Создайте рабочий каталог.
3. Дважды щёлкните кнопкой мыши на имени cab-файла. Будет выведен список файлов, упакованных внутри него.
4. Выделите все файлы, щёлкните правой кнопкой мыши и выполните команду **Извлечь (Extract)** открывшегося меню.
5. Перейдите в только что созданный рабочий каталог и щёлкните **Извлечь (Extract)**.
6. В окне Проводника Windows перейдите в рабочий каталог и откройте в Обозревателе Internet файл \_\_aaCabinetFileList.xml.

---

**Примечание.** Корневой узел в этом xml-файле – <aaCabinetFileList>. Этот узел состоит из xml-секций <FileItem>. Каждая из них содержит xml-атрибуты FileCode и FileName.

---

7. Найдите элемент FileItem со строкой "FileName=\*.Net.xml".
8. Создайте с помощью Блокнота Windows tmp-файл с названием, соответствующим значению FileCode в найденной на шаге 7 строке.
9. В xml-секции <dependentFiles> найдите xml-узел <file> с COM-объектом, который нуждается в регистрации. Измените значение элемента <registrationType> с "eNormal" на "eComDLL" (НЕ изменяется значение этого элемента для узла ??? .Net.dll).
10. Сохраните изменённый xml-файл.
11. Откройте окно командной строки и перейдите в рабочий каталог.
12. Введите команду "Cabarc N ??? .aaSLIB \*.\*", где символы "???" представляют собой имя модифицируемой библиотеки (как на шаге 1).

---

**Примечание.** Утилиту cabarc.exe можно переписать с веб-сайта Microsoft. Сохранить её следует в папке \system32.

---

Результирующий файл с расширением .aaSLIB может быть импортирован в Galaxu. Библиотека скриптовых функций будет автоматически разослана и зарегистрирована вместе с объектами, которые обращаются к содержащимся в ней функциям.

---

**Внимание!** Если COM-библиотека, разработанная с помощью Visual Studio версии 6 или более ранней, имеет зависимые файлы, их также следует включить в файл с расширением .aaSLIB и создать запись "<file> ... </file>" в xml-файле в секции <dependentFiles> для каждого такого файла. В противном случае правильность функционирования объектов, обращающихся к библиотеке скриптовых функций, не гарантируется.

---

---

**Внимание!** Для успешного повторного импортирования aaSLIB-файла нужно предварительно обновить сведения о версии в dll-библиотеке скриптовых функций. В противном случае dll-библиотека не будет должным образом зарегистрирована в качестве COM-объекта и не будет рассылаться вместе с объектом.

---

## Блокирование скриптов

Блокирование скриптов в шаблонах выполняется согласно следующим правилам:

- Имя скрипта и его существование неявным образом блокируется всегда. Это означает, что:
  - Скрипт не может быть удалён в порождённых объектах.
  - Имя скрипта не может быть изменено в порождённых объектах.
  - Изменение имени в данном шаблоне приводит к соответствующему переименованию во всех порождённых объектах.
  - После создания порождённых объектов удалить скрипт в базовом шаблоне возможно. При этом он также удаляется из порождённых объектов.
  - После создания порождённых объектов в шаблон можно добавить новый скрипт. При этом он также появится во всех порождённых объектах.
  - Новые скрипты можно добавлять в порождённые объекты без каких-либо ограничений.
- Текст скрипта в шаблоне может быть как заблокированным, так и не заблокированным. Возможные типы текстов: Declarations (объявления), Execute (исполнение), Startup (запуск), Shutdown (завершение), On Scan (включение сканирования) и Off Scan (выключение сканирования). Отдельно заблокировать некоторые из них в редакторе объектов невозможно. Групповая блокировка применяется или отменяется для всех типов текста одновременно.
  - После блокировки текста изменить его в порождённых шаблонах и экземплярах объекта невозможно.
- При блокировании текста в шаблоне автоматически блокируются и все имена-синонимы. Ссылки-синонимы никогда не блокируются. Блокирование синонимов отдельно не выполняется.
  - Блокирование синонимов означает блокирование всего списка имён-синонимов, включая количество элементов списка. Если список имён-синонимов заблокирован, добавить новые имена в списки производных шаблонов и экземпляров объектов невозможно. Ссылки-синонимы в производных шаблонах и экземплярах объектов модифицировать можно всегда, даже в тех случаях, когда список имён-синонимов будет заблокирован (это одна из основных функций имён-синонимов).
- Описание скрипта, флажок асинхронного исполнения, выражение, тип запуска, период запуска, допуск и аларм исполнения можно блокировать и разблокировать отдельно от текста скрипта. Для этого набора атрибутов предусмотрена групповая блокировка.
- После добавления нового скрипта все его параметры являются модифицируемыми.
- После добавления скрипта в экземпляр объекта модифицируемыми являются все его параметры, за исключением параметров блокировки (параметры блокировки в экземплярах объектов изменить невозможно).

---

**Внимание!** Как правило, в выражениях всегда используются ссылки на атрибуты. Чтобы заблокировать в шаблоне выражение и соответствующий скрипт, в них нужно использовать синонимы. Это позволит указывать



нужные атрибуты, которые определяются синонимами, для каждого экземпляра объекта в то время, когда код скрипта будет заблокирован.

Следующие правила применяются при создании производных объектов на основе шаблонов с заблокированными параметрами:

- Если параметр шаблона заблокирован, во всех производных шаблонах и экземплярах объектов значение соответствующего параметра будет равно значению заблокированного параметра. Изменить его можно только в том шаблоне, в котором он был заблокирован, после чего это изменение отразится во всех производных шаблонах и экземплярах объектов.
  - Блокирование параметра скрипта (с типом "текст" или "вид запуска") в шаблоне означает, что все производные шаблоны и экземпляры объектов будут указывать на этот заблокированный атрибут. Изменения значения заблокированного атрибута (например изменение текста) будет отражаться (и отображаться) во всех производных шаблонах и экземплярах объектов. По месту использования такие экземпляры объектов будут помечаться как находящиеся в состоянии обновления. После повторной пересылки в места использования изменение значения заблокированного атрибута шаблона будет выполнено также во всех используемых экземплярах объектов.
- Если параметр шаблона не заблокирован, соответствующие атрибуты всех производных шаблонов и экземпляров объектов могут иметь отличающиеся значения. Изменение значения незаблокированного атрибута в производных шаблонах и экземплярах объектов допускается. Изменение значения незаблокированного параметра в базовом шаблоне не оказывает никакого влияния на соответствующие параметры производных шаблонов и экземпляров объектов.
  - Отсутствие блокировки параметра скрипта (например выражения или порядка выполнения) означает, что во всех производных шаблонах и экземплярах объектов он может быть изменён произвольным образом, независимо от остальных. Изменение значения такого атрибута в базовом шаблоне (например модификация выражения) никак не сказывается на производных шаблонах и экземплярах объектов. Состояние используемых объектов при этом не изменяется. Повторная пересылка объектов не приводит к изменению значения соответствующих параметров.

## Определяемые пользователем атрибуты

На странице **Пользовательские атрибуты (UDAs)** окна редактора объектов возможно выполнение следующих действий:

- Определение нового атрибута объекта.
- Определение его типа.
- Определение категории атрибута.
- Указание начального значения и состояния блокировки атрибута.
- Определение нового атрибута как массива с указанием числа элементов.
- Разрешение генерации алармов и сохранения значений нового атрибута в архиве (в соответствии с определениями на странице **Расширения – Extensions**).

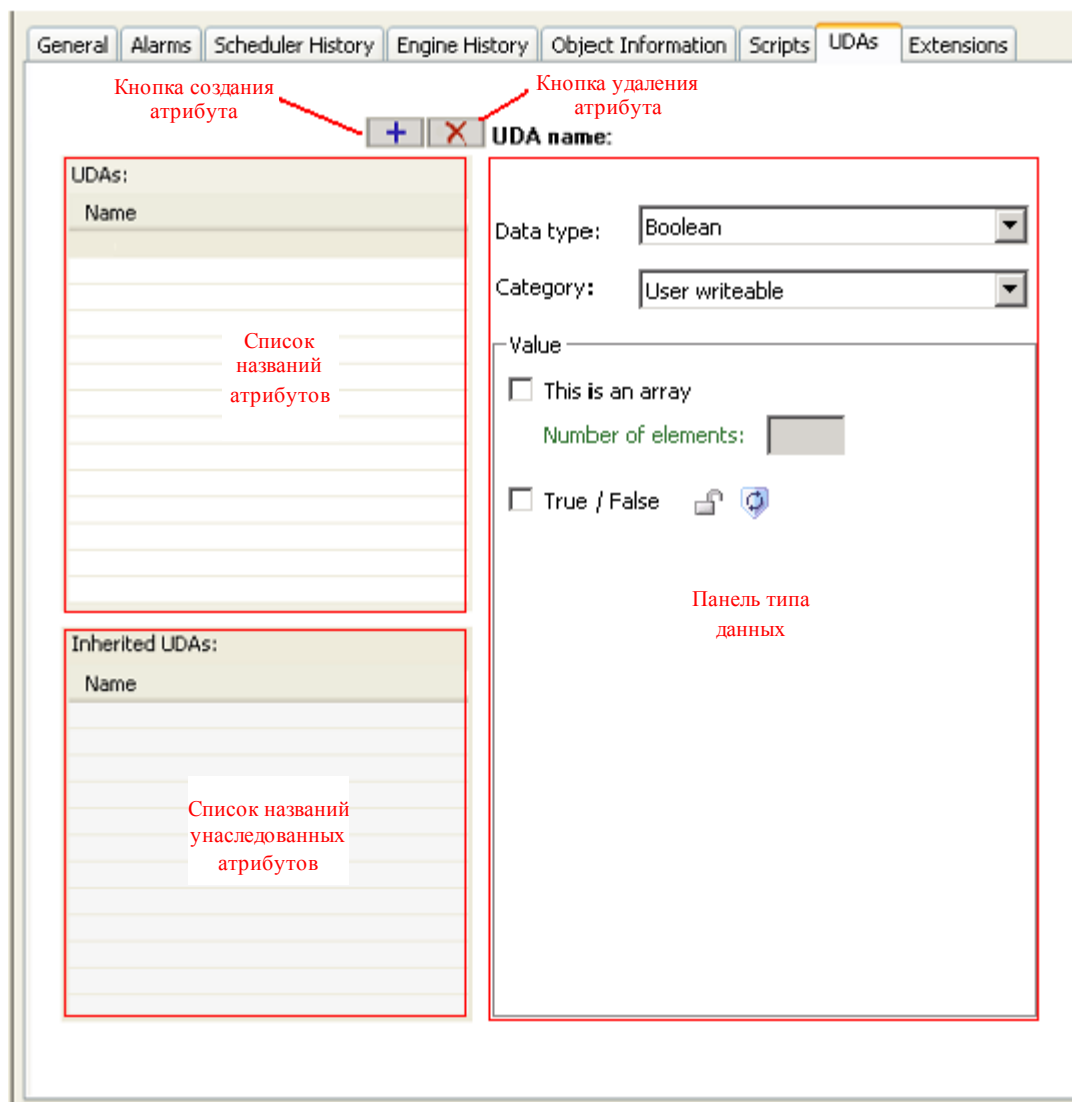
- Определение прав доступа и ссылок на другие объекты (в соответствии с определениями на странице **Расширения – Extensions**).

**Примечание.** Только что определённый атрибут сразу же появляется в списке атрибутов в окне браузера атрибутов и может быть использован в скриптах и в вызовах функций.

Пользовательский атрибут может быть определён как для шаблона, так и для экземпляра объекта. В первом случае атрибут, его тип данных и категория автоматически блокируются во всех экземплярах. Если в шаблоне будут заблокированы другие параметры (например начальное значение или определение прав доступа), значение пользовательского атрибута нельзя будет изменить ни в одном экземпляре объекта, созданного по этому шаблону. Если они в базовом шаблоне не заблокированы, их можно будет модифицировать и блокировать в производных шаблонах. После того как атрибут разблокирован в базовом или производном шаблоне, его значение можно изменять в экземплярах объектов.

## Страница Пользовательские атрибуты (UDAs)

На странице **Пользовательские атрибуты (UDAs)** имеются три основных функциональных области, а также набор функциональных кнопок.



Основные функциональные области страницы **Пользовательские атрибуты (UDAs)**:

- **Имена атрибутов (UDAs Name)**: перечень всех атрибутов, определённых в текущий момент для объекта.
- **Имена унаследованных атрибутов (UDAs Inherited Name)**: перечень всех атрибутов, унаследованных от родительского объекта.
- **Тип данных (Data Type)**: панель определения типа значений, которые может принимать атрибут.

На странице также имеются две кнопки:

- **Создать атрибут (Add UDA)**: при нажатии этой кнопки создаётся новый пользовательский атрибут объекта.
- **Удалить атрибут (Delete UDA)**: при нажатии этой кнопки выделенный элемент удаляется из списка атрибутов объекта.

Допустимыми типами значений атрибутов являются **Boolean** (логический), **Integer** (целый), **Float** (вещественный), **Double** (с двойной точностью), **String** (строковый), **Time** (время), **ElapsedTime** (прошедшее время), **InternationalizedString** (интернациональная строка).

Допустимые категории значений атрибута: **Calculated** (вычисляемый), **Calculated Retentive** (вычисляемый сохраняемый), **Object Writeable** (устанавливаемый объектом), **User Writeable** (устанавливаемый пользователем). Категории "Writeable" можно блокировать. При указании категории **Calculated** (вычисляемый) устанавливать значение атрибута могут только скрипты, исполняющиеся в этом же объекте.

Атрибуты всех типов, исключая **InternationalizedString** (интернациональная строка) могут определяться как массивы. Для этого нужно установить флажок **Это массив (This is an Array)** и указать в поле **Количество элементов (Number of Elements)** количество составляющих его элементов. При увеличении значения в этом поле количество строк в панели **Значение (Value)** будет также увеличиваться.

В панели **Значение (Value)** указываются начальные значения атрибута, которые он будет приобретать при пересылке объекта в место использования. Если атрибут представляет собой единичную логическую величину (не массив), установите флажок **Истина/Ложь (True/False)** в том случае, когда атрибут должен иметь значение "True", в противном случае сбросьте его. Если атрибут является массивом логических значений, укажите общее значение по умолчанию.

При использовании атрибутов в скриптах нужно принимать во внимание следующее:

- Если атрибуты **Calculated** (вычисляемый) или **Calculated Retentive** (вычисляемый сохраняемый) используются как счётчики, нужно явно определять их начальное значение. Например, если в скрипте выполняется оператор вида "me.UDA=me.UDA+1", до него должен быть выполнен оператор вида "me.UDA=1" или "me.UDA=значение\_другого\_атрибута".
- Начальное значение атрибутов **Calculated** (вычисляемый) может быть установлено в скриптах "OnScan" и "Execute" (то есть в скриптах с этим видом запуска), но не в скриптах "Startup".
- Начальное значение атрибутов **Calculated Retentive** (вычисляемый сохраняемый) должно устанавливаться в скриптах "Startup" и может устанавливаться в скриптах "OnScan" и "Execute". Основное назначение атрибутов данного типа заключается в том, чтобы сохранять текущее значение при повторной загрузке компьютера, при переключениях в резервируемых конфигурациях и в других аналогичных ситуациях, при которых создаётся допустимая

контрольная точка данных. Таким образом, в скрипте "Startup" должен присутствовать оператор, проверяющий логическое значение атрибута **StartingFromCheckpoint (начало с контрольной точки)** объекта AppEngine. Если оно равно "True", присваивать начальное значение атрибуту не нужно. Если оно равно "False", атрибуту нужно присвоить начальное значение.

Подробнее о типах скриптов см. параграф "Страница Скрипты (Scripts)".

### Чтобы создать пользовательский атрибут

1. Щёлкните символ "+" на странице **Пользовательские атрибуты (UDAs)** редактора объектов. В списке имён атрибутов появится новый элемент. При этом его имя будет отображаться в режиме изменения. Введите имя атрибута. (Чтобы удалить атрибут из списка, выделите его имя и щёлкните символ "x" красного цвета.)

---

**Внимание!** После создания пользовательского атрибута он (как и все остальные) тут же становится доступным на странице **Расширения (Extensions)** для дальнейшего расширения функциональных возможностей объекта. Если этот атрибут впоследствии будет удалён или переименован, все расширения объекта, определённые на странице **Расширения (Extensions)**, будут утеряны.

---

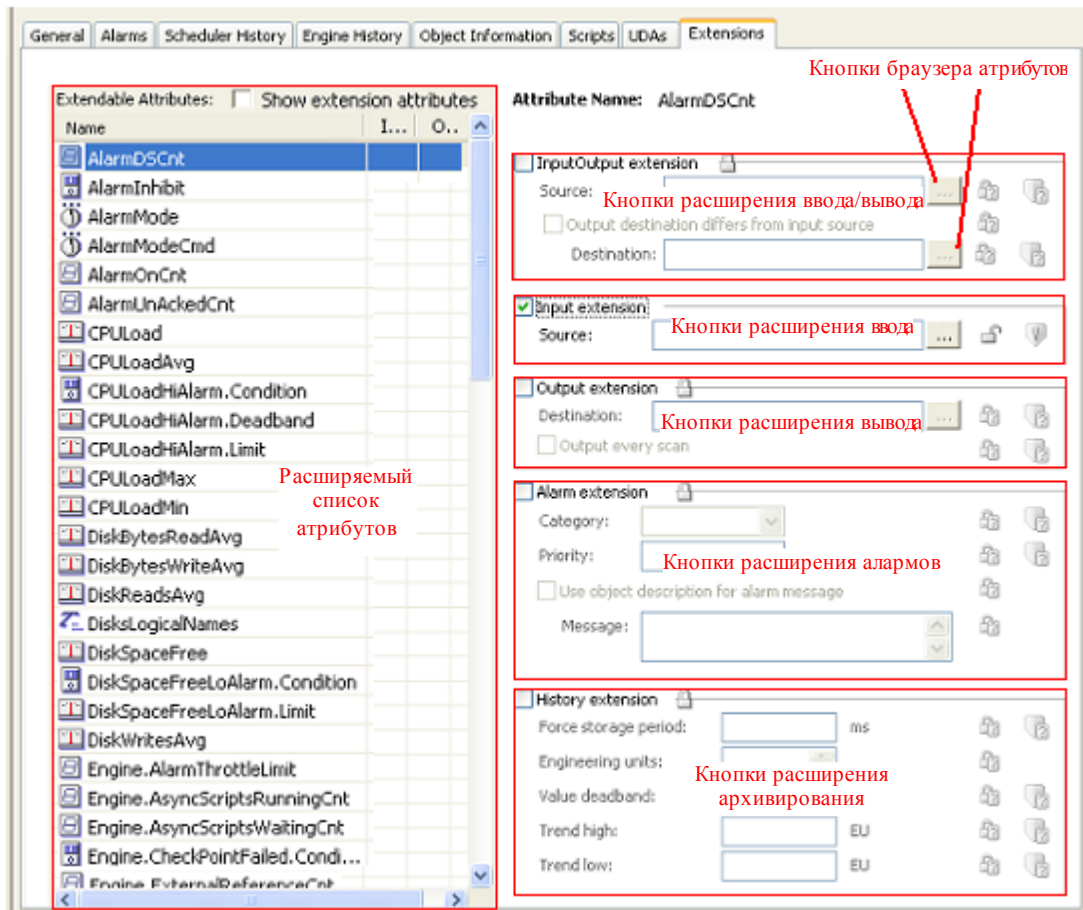
2. Укажите тип нового атрибута в поле **Тип данных (Data Type)**. Отображение остальных полей этой панели зависит от выбора конкретного типа данных.
3. Установите все значения нужным образом.
4. При необходимости заблокируёте значение атрибута. Кнопка блокировки будет доступна только при расширении шаблона. В остальных случаях этот знак обозначает состояние блокировки в родительском объекте. Если атрибут представляет собой массив, он также может быть заблокирован.
5. Укажите права доступа к атрибуту. Подробнее см. главу "Редакторы объектов".
6. Сохраните сделанные изменения и завершите работу редактора объектов, для того чтобы новые атрибуты появились в конфигурации объекта.

## Расширения атрибутов

На странице **Расширения (Extensions)** задаются характеристики существующих атрибутов, связанные с вводом, выводом, генерацией алармов и архивированием, не реализованные в исходном объекте.

### Страница Расширения (Extensions)

Страница расширений состоит из пяти функциональных областей, показанных на следующем рисунке.



Ниже описаны основные функциональные области страницы расширений атрибутов:

- **Расширяемый список атрибутов:** перечень всех определённых в текущий момент атрибутов объекта. Чтобы в этом списке отображались атрибуты, определённые на странице **Пользовательские атрибуты (UDAs)**, установите флажок **Показать пользовательские атрибуты (Show Extension Attributes)**.
- **Панель расширения в/в:** определение параметров считывания значения атрибута из внешнего источника и записи во внешний приёмник (которые могут как совпадать, так и быть разными). Считывание значений и качества данных атрибута **Источник (Source)** и обновление расширенного атрибута осуществляется в каждом периоде сканирования. Сведения, считанные из него, не записываются в атрибут **Приёмник (Destination)**.
- **Панель расширения ввода:** конфигурирование атрибута, который является ссылочным источником для другого объекта, то есть параметры получения расширенным атрибутом значений атрибута в поле **Источник (Source)**.
- **Панель расширения вывода:** параметры записи в атрибут другого объекта. Обновление атрибута в поле **Приёмник (Destination)** выполняется при изменении значения или качества значений расширенного атрибута (с "Bad" или "Initializing" на "Good" или "Uncertain").
- **Панель расширения алармов:** определение условий генерации алармов (когда логический атрибут принимает значение "True").

- **Панель расширения архивирования:** определение параметров архивирования значений атрибута, для которого эта функция ранее не выполнялась.

#### Чтобы определить расширенные характеристики атрибута

1. Выделите в списке на странице **Расширения (Extensions)** редактора объектов требуемый атрибут. Вид и состав полей четырёх остальных функциональных областей будет определяться типом выбранного атрибута.
2. Установите флажок желаемого типа расширения характеристик атрибута. При этом поля соответствующей панели станут доступными.
3. Укажите в панели расширений в/в атрибут-источник, введя его в поле ввода или нажав расположенную справа кнопку запуска браузера атрибутов. В появившемся окне браузера выберите нужный атрибут. Если атрибут-приёмник отличается от атрибута-источника, установите флажок **Приёмник отличается от источника (Output Destination Differs from Input Source)** и укажите атрибут-приёмник, введя его в поле ввода или нажав расположенную справа кнопку запуска браузера атрибутов. После определения расширенных характеристик в/в в столбце **Ввод/вывод (InputOutput)** списка имён атрибутов появится знак "X".

---

**Внимание!** Если сбросить флажок **Приёмник отличается от источника (Output Destination Differs from Input Source)**, в поле **Приёмник (Destination)** автоматически появится строка "---". В рабочей программе она будет интерпретироваться как ссылка на атрибут, указанный в поле **Источник (Source)** во время конфигурирования. Поскольку в рабочем приложении эту ссылку можно изменять, при сброшенном флажке **Приёмник отличается от источника (Output Destination Differs from Input Source)** блокировать параметр **Приёмник (Destination)** не следует.

---

4. В панели **Расширение ввода (Input Extension)** укажите атрибут-источник, введя его в поле ввода или нажав расположенную справа кнопку запуска браузера атрибутов. После определения расширенных характеристик ввода в столбце **Вход (Input)** списка имён атрибутов появится знак "X".
5. В панели **Расширение вывода (Output Extension)** укажите атрибут-приёмник, введя его в поле ввода или нажав расположенную справа кнопку запуска браузера атрибутов. Установите флажок **Выдавать в каждом периоде сканирования (Output Every Scan)**, если значение расширенного атрибута должно записываться в атрибут-приёмник в каждом периоде сканирования объекта (в противном случае оно будет записываться в атрибут-приёмник только при изменении значения или качества значения с "Bad" или "Initializing" на "Good" или "Uncertain"). После определения расширенных характеристик вывода в столбце **Output** списка имён атрибутов появится знак "X".
6. Укажите требуемые параметры в панели расширений алармов. В поле **Категория (Category)** выберите категорию алармов из следующего списка: **Discrete, Value LoLo, Value Lo, Value Hi, Value HiHi, DeviationMinor, DeviationMajor, ROC Lo, ROC Hi, SPC, Process, System, Batch, Software**. Укажите приоритет аларма в поле **Приоритет (Priority)**. По умолчанию он равен 500. Установите флажок **Использовать описание объекта в качестве сообщения аларма (Use Object Description for Alarm Message)** или введите текст другого сообщения об аларме в поле **Сообщение (Message)**. После определения расширенных характеристик генерации алармов в столбце **Аларм (Alarm)** списка имён атрибутов появится знак "X".

7. Укажите в панели **Расширение архивирования (History Extension)**, в зависимости от типа данных выбранного атрибута, значения оставшихся параметров: **Периодичность сохранения (Force Storage Period)**, **Единицы измерения (Engineering Units)**, **Допуск на значения (Value Deadband)**, **Максимальное значение тренда (Trend High)** и **Минимальное значение тренда (Trend Low)**. После определения расширенных характеристик архивирования в столбце **История (History)** списка имён атрибутов появится знак "X".
8. При необходимости заблокируйте характеристики атрибута. Кнопка блокировки будет доступна только при расширении шаблона. В других случаях этот знак отображает состояние блокировки в родительском объекте.
9. Укажите требуемые права доступа. Подробнее см. главу "Редакторы объектов".
10. Сохраните сделанные изменения и завершите работу редактора объектов, для того чтобы новые характеристики атрибутов вступили в силу.

## Расширенная характеристика в/в InputOutput

Назначение этой панели – указать, из какого внешнего атрибута может быть получено значение текущего атрибута объекта (шаблона или экземпляра объекта) и в какой внешний атрибут оно может быть записано. Цель этого определения заключается в обеспечении возможности мониторинга значений или качества значений входных величин и выдачи сигнала об их изменении.

Атрибут-приёмник может как отличаться от атрибута-источника, так и совпадать с ним. Ссылки всегда делаются на атрибуты совпадающего типа.

Для каждого объекта AutomationObject может быть определено несколько расширенных атрибутов с характеристикой InputOutput. Вместе с тем определить эту характеристику для атрибута, который уже имеет характеристики Input или Output, невозможно.

---

**Примечание.** Определение характеристики InputOutput для блокируемых атрибутов возможно, однако в рабочем приложении правильность использования таких атрибутов будет обеспечена только тогда, когда они будут незаблокированными.

---

Когда сканирование объекта разрешено, значение и качество значения атрибута с расширенной характеристикой InputOutput отражает значение и качество значения внешнего атрибута (при успешном считывании). Если считывание завершается с ошибкой (например в случае сбоя канала связи или ошибок преобразования данных), индикатор качества значения расширенного атрибута устанавливается в "Bad".

В режиме сканирования действия с расширенным объектом выполняются следующим образом: если внешняя операция (например команда пользователя) с расширенным атрибутом приводит к изменению его значения или качества, операция записи значения расширенного атрибута в атрибут-приёмник будет выполнена в течение следующего шага исполнения. При этом нужно, чтобы индикатор качества имел значение "Good" или "Uncertain". Для записи в атрибут-приёмник вследствие изменения качества нужно, чтобы индикатор качества изменил своё значение с "Bad" или "Initializing" на "Good" или "Uncertain". Важную роль в контроле выходных значений имеет атрибут с названием WriteValue, который является открытым для общего доступа.

Когда сканирование объекта не осуществляется, пользовательские команды установки значения принимаются, но индикатор качества при этом всегда получает значение "Bad".

Атрибут с характеристикой `InputOutput` используется в скриптах двояким образом, в зависимости от того, выполняется ли действие с входным значением или с выходным.

В первом случае скрипт получает значение из атрибута-источника и производит над ним некоторые действия. В операторе при этом должна быть указана ссылка непосредственно на атрибут. Например, если расширенным атрибутом является атрибут `"me.udal"`, в операторах и выражениях скрипта должно быть записано `"me.udal"`.

Во втором случае скрипт выводит новое значение или проверяет его. В операторах и выражениях скрипта должна быть при этом указана ссылка на атрибут `"WriteValue"` расширенного атрибута (например `"me.udal.WriteValue"`). Таким образом, для того чтобы проверить новое значение атрибута `"udal"`, в операторе сравнения нужно написать выражение `"me.udal.WriteValue"`. Если выходная величина должна иметь конкретное значение, в операторе присваивания также должна быть такая же строка. Например:

```
IF (me.udal.WriteValue > 100.0) THEN
    me.udal.WriteValue = 100.0;
ENDIF;
```

Данные должны быть записаны в атрибут `"me.udal.WriteValue"`, поскольку именно эта величина будет изменяться перед фактической записью нового значения в выходное поле. В скрипте может быть выполнен перехват этого значения непосредственно перед операцией записи и его изменения. Чтобы значение атрибута `WriteValue` не передавалось в выходное поле, индикатор качества нужно установить в `"Bad"` с помощью функции `SetBad()`.

Подробнее см. параграф "Выполнение операций вывода".

## Расширенная характеристика ввода Input

Расширенная характеристика ввода может быть определена только для атрибутов, запись в которые возможна. Массивы не поддерживаются.

Если типы расширенного атрибута и атрибута-источника совпадают, они будут получать идентичные значения с периодичностью исполнения программного обеспечения расширенного объекта. Если их типы разные, будет выполняться преобразование данных. В случае ошибки преобразования или выхода входного значения за пределы допустимых значений расширенного атрибута индикатор качества данных расширенного атрибута устанавливается в `"Bad"`. В противном случае индикатор качества данных расширенного атрибута получает значение индикатора качества данных атрибута-источника.

Когда сканирование объекта не осуществляется, пользовательские команды установки значения принимаются, но индикатор качества при этом всегда получает значение `"Bad"`.

Полномочия пользователей для выполнения действий с расширенным атрибутом с характеристикой `Input` не проверяются. Выполняется только проверка того, имеет ли пользователь ИСР право модифицировать объект (с целью его расширения).

Данная характеристика может быть определена как для атрибута шаблона, так и для атрибута экземпляра объекта. В первом случае удалить её в производных объектах будет нельзя.

## Расширенная характеристика вывода Output

Атрибуты, запись в которые возможна, и атрибуты "только для чтения" могут быть расширены характеристикой вывода. Массивы не поддерживаются.



Если типы данных расширенного атрибута и атрибута-приёмника совпадают, оба атрибута получают одинаковые значения тогда, когда индикатор качества значения расширенного атрибута равен "Good". Если типы разные, – выполняется преобразование данных. В случае ошибки преобразования система генерирует ошибку конфигурирования и несоответствия типов.

Данная характеристика может быть определена как для атрибута шаблона, так и для атрибута экземпляра объекта. В первом случае удалить её в производных объектах нельзя. Выходной атрибут-приёмник можно блокировать в шаблонах отдельно.

Подробнее см. параграф "Выполнение операции вывода".

---

**Примечание.** Свойство "quality" атрибута, расширенного путём добавления характеристики Output, обладает следующим действием: значения могут быть записаны в атрибут-источник только тогда, когда индикатор качества имеет значение "Bad" или "Uncertain". Сведения о качестве не передаются (только значение), поскольку это свойство атрибута не выводится при выполнении операций установки. Если индикатор качества изменяется с "Bad" или "Initializing" на "Good" или "Uncertain", значение выводится, даже если оно не изменилось. Если индикатор качества изменяется с "Good" на "Uncertain", при этом само значение не изменилось, значение не выводится. Когда сканирование объекта прекращается, вывод данных не осуществляется. Когда сканирование расширенного объекта не осуществляется, пользовательские команды установки значения принимаются, а индикатор качества при этом всегда получает значение "Good".

---

## Расширенная характеристика алармов Alarm

Данная характеристика может быть определена как для атрибута шаблона, так и для атрибута экземпляра объекта. В первом случае удалить её в производных объектах нельзя.

Атрибуты, являющиеся массивами, расширять нельзя.

## Расширенная характеристика архивирования History

Архивирование может быть определено для любого атрибута, значения которого ещё не записываются в архив и который существует у объекта в функционирующем приложении.

Эта характеристика может быть определена для атрибутов, запись в которые возможна, и для атрибутов "только для чтения" следующих типов: Float (вещественный), Double (с двойной точностью, сохраняется как вещественный), Integer (целый), Boolean (логический), String (символьный, сохраняется как строки в кодировке Unicode длиной не более 512 символов), CustomEnumeration (перечислительный, сохраняется как целый), ElapsedTime (прошедшее время, сохраняется как секунды).

В панели расширения архивирования имеются следующие поля:

- **Периодичность сохранения (Force Storage Period):** интервалы времени, через которые система должна записывать данные в архив, даже если их значение не изменилось.
- **Единицы измерения (Engineering Units):** единицы измерения значений архивируемого атрибута.
- **Допуск на значения (Value Deadband):** отклонение (выраженное в единицах измерения), на которое новое значение должно отличаться от последнего сохранённого, для того чтобы быть записанным в архив. "0" допускается и означает, что архивирование будет выполнено при

любом изменении значения данных. Изменение качества данных всегда приводит к созданию новой записи архива, независимо от изменения значений данных.

- **Максимальное значение тренда (Trend High):** максимальное значение шкалы тренда по умолчанию.
- **Минимальное значение тренда (Trend Low):** минимальное значение шкалы тренда по умолчанию.

Данная характеристика может быть определена как для атрибута шаблона, так и для атрибута экземпляра объекта. В первом случае удалить её в производных объектах будет нельзя.

## Выполнение операций вывода

Следующая информация действительна для характеристик InputOutput и Output, а также функций вывода в объектах Field Reference, Switch и Analog Device.

При выдаче единственного запроса на установку значения атрибута-приёмника в одном периоде сканирования в пункт назначения передаётся указанная величина. Вместе с тем в одном и том же периоде сканирования возможна выдача нескольких запросов на установку значений. В этом случае в пункт назначения передаётся последняя из указанных величин.

Порядок выполнения операции вывода следующий: получив управление в текущем периоде сканирования, объект пересылает в атрибут назначения последнюю из указанных величин. Статус объекта изменяется на "Pending" (ждущий), так как он ожидает подтверждения записи от объекта назначения. Все остальные запросы в этом периоде сканирования помечаются как успешно выполненные.

Если в следующем периоде сканирования будут выданы новые запросы на установку, передаваемое значение будет определено таким же образом, как и для первого цикла. Оно пересылается в пункт назначения, после чего операция вывода предшествующего периода сканирования помечается как успешно завершённая, даже если объект в действительности не получил подтверждения о выполненной объектом назначения записи в атрибут-приёмник.

Например, если в текущем периоде были выданы последовательные запросы на установку значений {11,24,35,35,22,36,40}, объекту назначения будет передано значение последнего запроса (40), а все остальные будут помечены как успешно выполненные.

Исключение составляет обработка данных логического типа, указываемых в пользовательских запросах на установку (от приложений InTouch или FactorySuite Gateway). В ней учитывается как случайная частота ввода данных пользователем (например повторяющиеся нажатия на пусковую кнопку), так и постоянная частота сканирования выходов объекта, что позволяет получать воспроизводимые результаты. В данном случае описанная свёртка данных и поддержание очереди в один элемент позволяет лучше обслуживать запросы пользователей. Самое первое значение, указанное после передачи объекта в место использования (значение по умолчанию "True" или "False"), всегда передаётся в атрибут-приёмник.

Впоследствии в периоде сканирования применяется двухуровневая схема кэширования на базе очередей "Передаваемых значений" и "Следующих передаваемых значений". Передаваемое значение определяется по результатам сравнения текущего значения и последнего переданного приёмнику значения. Следующее передаваемое значение определяется по результатам сравнения с передаваемым значением. При первом же изменении данных новое значение ставится в очередь передаваемых значений. Свёртка данных осуществляется в случае, когда в очередном

запросе указана та же самая величина. Следующее изменившееся значение ставится уже в очередь следующих передаваемых значений, свёртка выполняется аналогичным образом.

В очередном цикле сканирования передаются данные из очереди передаваемых значений, в следующем – данные из очереди следующих передаваемых значений.

Описанный механизм демонстрирует следующий пример:

Значение, переданное в предшествующем периоде сканирования	Запросы на установку, выданные в этом периоде сканирования	Передаваемое значение	Следующее передаваемое значение
0	1,0,0,1,1	1	-
1	1,0,0,1,1	0	1
0	1,1,0,0	1	0
1	1,1,0,0	0	-

**Примечание.** При обработке логических данных в запросах Супервизора (пересылки между объектами ApplicationObject и ArchestrA) или в комбинации запросов Супервизора и пользователя в одном и том же периоде сканирования их обработка выполняется таким же образом, как и обработка данных других типов.

**Внимание!** Если один и тот же атрибут расширяется характеристиками Input (вход) и Output (выход), запись в приёмник, определяемый характеристикой вывода, производится в каждом периоде сканирования, независимо от изменения расширенного атрибута. Запись будет выполняться даже при сброшенном флажке **Output Every Scan (выводить каждый период сканирования)**, что может привести к дополнительному сетевому трафику. При расширении атрибута характеристикой InputOutput данное действие выполняться не будет.

## Язык скриптов QuickScript .NET

В настоящем параграфе приведены примеры стандартных конструкций скриптов, а также описания функций, входящих в состав IAS по умолчанию. Все функции сгруппированы в виде библиотек, включая те, которые используются наиболее часто.

В синтаксисе языка QuickScript .NET поддерживаются следующие конструкции .NET:

- Объявление .NET-переменных.
- Вызов открытых .NET-конструкторов (как с аргументами, так и без аргументов).
- Вызов открытых .NET-методов экземпляра (как с аргументами, так и без аргументов).
- Вызов открытых статических .NET-методов (как с аргументами, так и без аргументов).
- Вызов открытых .NET-индексаторов (с одним аргументом и более).
- Использование .NET-перечислителей.

Следующие выражения являются ключевыми словами языка QuickScript .NET:

and	exit	null
as	false	Object
attribute	float	or
Boolean	for	real
dim	if	shl
discrete	in	shr
double	indirect	step
each	integer	string
elapsedtime	message	then
else	mod	time
elseif	new	to
endif	next	true
endwhile	not	while

## Ключевое слово Indirect

Ключевое слово Indirect предназначено для динамической привязки переменных к произвольным строкам.

Примеры использования ключевого слова Indirect, а также соответствующего метода BindTo() показаны ниже.

```
DIM x AS indirect;
```

```
...
```

```
x.BindTo(s); ' где "s" – любое выражение, возвращающее результат в вид строки
```

```
x = 1234;
```

```
IF WriteStatus(x) == MxStatusOk THEN
```

```
' ... какие-либо действия
```

```
endif;
```

---

**Внимание!** Указание аргументов в обращении **Attribute()** не поддерживается. Динамические ссылки на значения атрибутов создаются с помощью механизма Indirect.

---

Следующие слова зарезервированы для будущего использования:

abstract	error	me	switch
base	event	namespace	this
begin	explicit	noting	throw
break	extern	on	trigger
call	finally	out	try
case	function	private	typeof
catch	goto	protected	until
class	implicit	public	using
continue	import	raiseevent	virtual
default	imports	readonly	void
delegate	include	ref	when
do	inherits	return	with
end	interface	select	

endsub	internal	sizeof	
endswitch	is	static	
enum	like	sub	

**Примечание.** Написание ключевых слов и выражений QuickScript .NET, а также имён переменных нечувствительно к регистру используемых символов. Регистр символов сохраняется при работе в редакторе объектов.

## Общие правила разработки скриптов

В настоящем параграфе описаны общепринятые стили, синтаксис, команды и характеристики среды разработки скриптов.

Комментарии в тексте скриптов могут быть как одно-, так и многострочными. Однострочные комментарии в исходном тексте должны начинаться с символа одинарной кавычки ( " ' " ) и не требуют указания завершающей одинарной кавычки. Многострочные комментарии должны быть заключены между символами открывающей и закрывающей скобок " { " и " } " .

Для форматирования текста могут применяться пробелы, символы табуляции и пустые строки. Отступ в строке следует задавать с помощью клавиши TAB. Отдельные операторы должны отделяться друг от друга символом точки с запятой, обозначающим конец оператора.

## Типы и синтаксис скриптов

Условно скрипты могут быть разделены на две категории: простые и сложные. В простых скриптах используются операторы присваивания и сравнения, несложные математические функции и другие аналогичные операторы. В сложных скриптах применяются различные логические операторы, такие как IF-THEN-ELSE. В следующих описаниях рассматриваются простые операторы. Более подробные сведения о сложных скриптах см. параграф "Управляющие структуры QuickScript .NET".

## Поддерживаемый синтаксис выражений и операторов

Синтаксис используемых в скриптах выражений аналогичен синтаксису операций для калькулятора. Большинство выражений представляют собой операторы присваивания наподобие следующего:

$$a = (b - c) / (2 + x) * xyz;$$

При выполнении этого оператора результат вычислений с переменными справа от оператора присваивания ("=") записывается в переменную, указанную слева от него (в данном случае в переменную "a"). Все операторы должны заканчиваться точкой с запятой (";"). В качестве операндов в выражениях могут использоваться как константы, так и переменные. Слева от оператора присваивания должен стоять только один операнд.

Операнды могут объединяться знаком "+". Например, при каждом выполнении следующего скрипта, запускаемого при изменении данных, имя объекта "Setpoint" может изменяться соответствующим образом.

```
Setpoint.Name = "Setpoint" + Text(Number, "#");
```

Если переменная Number будет иметь значение 1, объекту Setpoint будет присвоено название "Setpoint1".

## Пример простого скрипта

В простых скриптах, как правило, используются операторы сравнения и несложные математические функции и операции. Например:

```
React_Temp = 150;
ResultTag = (Sample1 + Sample2) / 2;
{... ..}
```

## Операторы языка QuickScript .NET

В следующей таблице перечислены унарные операторы QuickScript .NET (то есть требующие указания только одного оператора).

Оператор	Значение
~	дополнительный код
-	отрицание
NOT	логическое НЕ

Операторы QuickScript .NET, требующие указания двух операндов:

Оператор	Значение
*	умножение
/	деление
+	сложение и конкатенация
-	вычитание
+	присвоение
MOD	деление по модулю
SHL	сдвиг влево
SHR	сдвиг вправо
&	поразрядное И
^	исключающее ИЛИ
	включающее ИЛИ
**	возведение в степень
<	меньше чем
>	больше чем
<=	меньше чем или равно
>=	больше чем или равно
==	эквивалентно; не используется для сравнения массивов (массивы должны сравниваться поэлементно).
<>	не равно
AND	логическое И
OR	логическое ИЛИ

Таблица приоритетов операторов:

Приоритет	Операторы
1 (наивысший)	()
2	- (отрицание), NOT, ~
3	**
4	*, /, MOD

5	+, - (вычитание)
6	SHL, SHR
7	<, >, <=, >=
8	==, <>
9	&
10	^
11	
12	=
13	AND
14 (самый низкий)	OR

В качестве операндов указанных операторов могут выступать как числа, так и значения атрибутов. Употребление круглых скобок для отделения аргументов необязательно. Операторы могут записываться в любом регистре.

### Круглые скобки

Круглые скобки используются для задания порядка вычислений, отличного от порядка задаваемого приоритетами операторов. Кроме того, их использование облегчает понимание сложных выражений. Действия над операндами в скобках выполняются раньше остальных. В следующем примере указано, что сперва нужно вычислить сумму переменных В и С, а затем результат сложения умножить на значение переменной D:

$$(B + C) * D$$

### Отрицание

Оператор отрицания является унарным и означает изменение знака операнда (например перевод положительного числа или значения переменной в отрицательное число и наоборот).

### Дополнительный код

Этот также унарный оператор, вычисляющий дополнение 32-разрядного целого числа до единицы (то есть он выполняет инвертирование каждого разряда: "0" заменяет на "1", а "1" – на "0").

### Возведение в степень

Данный унарный оператор возвращает результат возведения числа (основания) в указанную степень. Основание и показатель степени могут выражаться как целыми, так и вещественными значениями со следующими ограничениями:

Нулевое основание и отрицательный показатель степени недопустимы.

Пример: "0\*\*-2" и "0\*\*-2.5".

Отрицательное основание и не целый показатель степени также недопустимы.

Пример: "-2\*\*2.5" и "-2\*\*-2.5".

Ошибочное указание операторов даёт нулевой результат. Кроме того, результат выполнения операции не должен быть слишком большим или слишком маленьким, то есть таким, который не может быть представлено в виде вещественного числа.

Примеры операторов возведения в степень:

$$1 ** 1 = 1.0$$
$$5 ** 2 = 25.0$$

```
10 ** 5 = 100000.0
```

### Умножение, деление, сложение и вычитание

Бинарные операторы, выполняющие основные арифметические действия. Оператор сложения ("+") может использоваться для конкатенации (объединения) значений Message (сообщение) или Memory (память). Например, при каждом изменении значения переменной Number значение переменной Setpoint в результате выполнения следующего оператора будет также изменяться.

```
Setpoint.Name = "Setpoint" + Text (Number, "#");
```

Если переменная Number имеет значение 3, результатом выполнения оператора будет "Setpoint3".

### Деление по модулю

Данный оператор вычисляет остаток деления одного целого числа (записываемого слева от оператора) на другое (записываемое справа).

Пример:

```
97 MOD 8 = 1
```

```
63 MOD 5 = 3
```

### Сдвиг влево (SHL), сдвиг вправо (SHR)

Операторы SHL и SHR представляют собой бинарные операторы, аргументом которых могут быть числа только целого типа. Двоичные разряды целого числа (записываемого слева от оператора) поразрядно сдвигаются влево или вправо на количество позиций, указываемых вторым целым числом (записываемым справа). Разряды, выходящие за границу слова, теряются. Освобождающиеся позиции заполняются нулями. Данная операция представляет собой сдвиг без сохранения знакового разряда.

### Поразрядное И

Этот бинарный оператор выполняет логические операции И с соответствующими разрядами заданных 32-разрядных значений целого типа. Как правило, он используется для "маскирования" некоторых разрядов числа. Например, при выполнении следующего оператора старшие 24 разряда 32-разрядного числа будут "маскированы" (сброшены в 0), а младшие восемь будут сохранены:

```
Result = name & 0xff;
```

### Исключающее ИЛИ, включающее ИЛИ

Эти бинарные операторы выполняют поразрядное сравнение заданных 32-разрядных значений целого типа. Результат исключающего ИЛИ вычисляется следующим образом: если значения разрядов операндов совпадают, соответствующий разряд результата устанавливается в "0", в противном случае он устанавливается в "1". То есть:

```
0 ^ 0 = 0
```

```
0 ^ 1 = 1
```

```
1 ^ 0 = 1
```

```
1 ^ 1 = 0
```

Результат выполнения включающего ИЛИ определяется следующим образом: если хотя бы один из разрядов обоих операндов равен "1", соответствующий разряд результата также устанавливается в "1", в противном случае он устанавливается в "0":

```
0 | 0 = 0
```

```
0 | 1 = 1
```

```
1 | 0 = 1
```



1 | 1 = 1

### Оператор присваивания

Операнды этого бинарного оператора могут быть как целого, так и вещественного типа. В операторе может использоваться только один знак присваивания ("="). С левой стороны оператора может указываться только одно имя переменной. Знак равенства в данном случае следует читать как "получает значение (результат)".

---

**Примечание.** Функцию равенства выполняет знак эквивалентности ("=="), указываемый в логическом выражении конструкции IF-THEN-ELSE.

---

### Операторы сравнения

Эти операторы используются в логическом выражении конструкции IF-THEN-ELSE для определения того, какой из указанных в этой конструкции наборов операторов должен выполняться.

### Операторы AND, OR и NOT

Операнды этих операторов могут быть только логического типа. Если в качестве аргументов указаны значения целого или вещественного типа, они будут преобразованы следующим образом:

- Вещественное в логическое: если значение вещественного значения равно 0,0, оно трактуется как логический "0", в противном случае – как логическая "1".
- Целое в логическое: если значение целого значения равно 0, оно трактуется как логический "0", в противном случае – как логическая "1".

Например, если значение переменной Real1 равно 23,7, значение переменной Real2 равно 0,0, в результате выполнения оператора

```
Disc1 = Real1 AND Real2;
```

переменная Disc1 получит значение "0", так как значению Real1 соответствует логическая "1", а значению Real2 – логический "0":

---

**Внимание!** При присваивании переменной целого типа результата выполнения математической операции IAS выполняет округление результата до ближайшего целого, а не отбрасывание дробной части. Таким образом, при выполнении оператора "IntAtr = 32/60" переменная IntAtr получит значение "1", а не "0". Отбрасывание дробной части выполняется функцией Trunc().

---

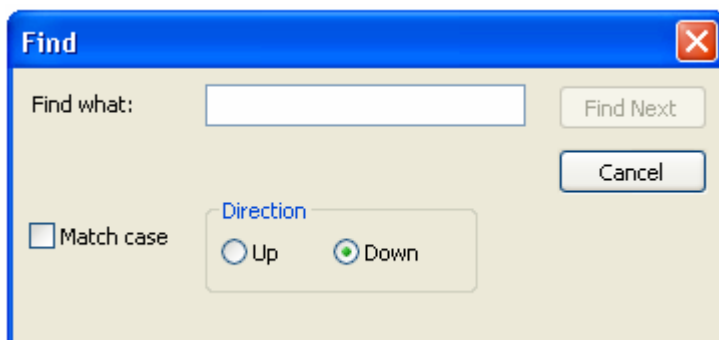
## Меню правой кнопки в окне создания скриптов

В состав меню, которое отображается при щелчке правой кнопкой мыши в окне создания скриптов, входят следующие команды:

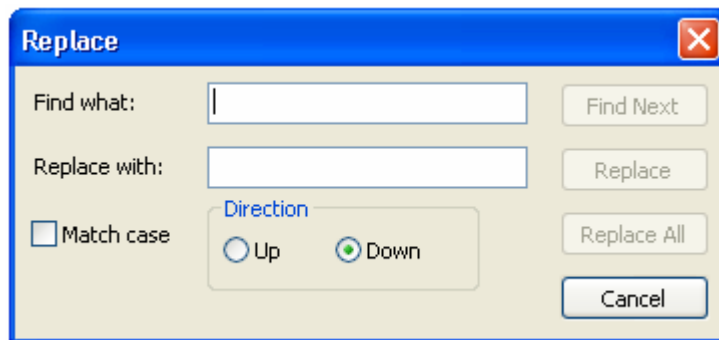
Команда	Описание
Cut (Вырезать)	Вырезание выделенного текста в Буфер обмена Windows.
Copy (Копировать)	Копирование выделенного текста в Буфер обмена Windows.
Paste (Вставить)	Вставка содержимого Буфера обмена Windows в позицию курсора.
Delete (Удалить)	Удаление выделенного текста без копирования его в Буфер обмена Windows.
Select All (Выбрать всё)	Выделение всего текста в окне создания скриптов.
Zoom In	Увеличение размера шрифта текста в окне создания

(Увеличить)	скриптов.
Zoom Out (Уменьшить)	Уменьшение размера шрифта текста в окне создания скриптов.
Find (Найти)	Команда поиска указанной последовательности символов в тексте. При её выполнении появляется показанное ниже окно <b>Найти (Find)</b> , в котором указывается искомая строка.
Find Next (Найти далее)	Команда поиска следующего вхождения строки символов, указанной при выполнении команды <b>Find</b> .
Replace (Заменить)	Команда поиска указанной последовательности символов и замены её на другую последовательность символов. При её выполнении появляется окно <b>Replace (Заменить)</b> , показанное ниже.
Go To (Перейти)	Команда перемещения курсора к строке исходного текста с указанным номером. При её выполнении появляется окно <b>Go To Line (Перейти к строке)</b> , показанное ниже.

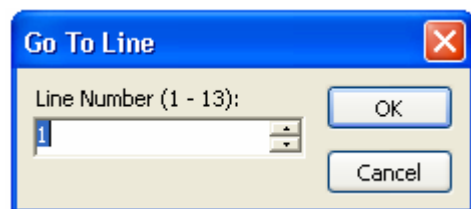
Окно поиска:



Окно замены:



Окно перехода:



Чтобы распечатать текст скрипта, выполните команды **Select All (Выделить всё)**, **Copy (Копировать)**, затем запустите программу текстового редактора, выполните в нём команду **Paste (Вставить)** и распечатайте вставленный текст средствами этого редактора.

## Скриптовые функции

В данном параграфе описаны все функции, которые по умолчанию доступны в среде разработки IAS. Их перечень дан в алфавитном порядке. Кроме того, приведены назначение, категория, синтаксис, описание аргументов и примеры использования функции.

### Abs()

Возвращает абсолютное значение (модуль) указанного числа или выражения.

#### Категория

Math (математические).

#### Синтаксис

*Result* = Abs (*Number*) ;

#### Аргументы

*Number*                    любое число или числовое выражение.

#### Примеры использования

Abs (14) ; ' ..... 14

Abs (-7.5) ; ' ..... 7.5

### ActivateApp()

Возобновляет исполнение другого запущенного ранее приложения Windows.

#### Категория

Miscellaneous (Разные).

#### Синтаксис

ActivateApp (*TaskName*)

#### Аргументы

*TaskName*                    название активизируемой задачи.

#### Примечание

Аргумент *TaskName* является текстовой строкой, в которой должно быть указано название задачи в таком же виде, в каком оно отображается в Панели задач Windows или в Менеджере задач, включая пробелы. (Менеджер задач запускается щелчком правой кнопки мыши на Панели задач Windows и выполнения команды **Менеджер задач – Task Manager** открывшегося меню или путём нажатия сочетания клавиш Ctrl+Alt+Del).

#### Примеры использования

ActivateApp ("Calculator") ;

### ArcCos()

Возвращает значение угла из диапазона от 0 до 180 градусов, косинус которого равен значению аргумента.

#### Категория

Math (Математические).

#### Синтаксис

*Result* = ArcCos (*Number*)

**Аргументы**

*Number* число или числовое выражение, значение которого находится в диапазоне от -1 до +1 (включительно).

**Примеры использования**

`ArcCos(1)` возвращает 0

`ArcCos(-1)` возвращает 180

**См. также**

`Cos()`, `Sin()`, `Tan()`, `ArcSin()`, `ArcTan()`

**ArcSin()**

Возвращает значение угла из диапазона от -90 до +90 градусов, синус которого равен значению аргумента.

**Категория**

Math (Математические).

**Синтаксис**

*Result* = `ArcSin(Number)`

**Аргументы**

*Number* число или числовое выражение, значение которого находится в диапазоне от -1 до +1 (включительно).

**Примеры использования**

`ArcSin(1)` ..... 90

`ArcSin(-1)` ..... -90

**См. также**

`Cos()`, `Sin()`, `Tan()`, `ArcCos()`, `ArcTan()`

**ArcTan()**

Возвращает значение угла из диапазона от -90 до +90 градусов, тангенс которого равен значению аргумента.

**Категория**

Math (Математические).

**Синтаксис**

*Result* = `ArcTan(Number)`

**Аргументы**

*Number* число или числовое выражение.

**Примеры использования**

`ArcTan(1)` возвращает 45

`ArcTan(0)` возвращает 0

**См. также**

`Cos()`, `Sin()`, `Tan()`, `ArcCos()`, `ArcSin()`

**Cos()**

Возвращает значение косинуса угла, выраженного в градусах.

**Категория**

Math (Математические).

#### Синтаксис

*Result* = *Cos* (*Number*)

#### Аргументы

*Number*          число или числовое выражение.

#### Примеры использования

*Cos* (90) возвращает 0

*Cos* (0) возвращает 1

Ниже показан пример использования функции в математическом выражении:

```
Wave = 50 * Cos (6 * Now() .Second) ;
```

#### См. также

*Sin*(), *Tan*(), *ArcCos*(), *ArcSin*(), *ArcTan*()

## CreateObject()

Функция создания объекта ActiveX (COM-объекта).

#### Категория

System (Системные).

#### Синтаксис

*ObjectResult* = *CreateObject* (*ProgID*) ;

#### Аргументы

*ProgID*          программный идентификатор (в виде строки символов) создаваемого объекта.

#### Примеры использования

```
CreateObject ("ADODB.Connection") ;
```

## DText()

Возвращает одну из символьных строк в зависимости от значения логического аргумента.

#### Категория

String (Символьные).

#### Синтаксис

*StringResult* = *DText* (*Discrete*, *OnMsg*, *OffMsg*) ;

#### Аргументы

*Discrete*        логическое значение или выражение логического типа;

*OnMsg*          строка символов, которая будет отображаться, если значение аргумента *Discrete* равно "True";

*OffMsg*         строка символов, которая будет отображаться, если значение аргумента *Discrete* равно "False".

#### Примеры использования

```
StringResult = DText (me.temp > 150, "Too hot", "Just right") ;
```

## Exp()

Возвращает результат возведения числа  $e$  в указанную степень.

### Категория

Math (Математические).

### Синтаксис

```
Result = Exp(Number);
```

### Аргументы

*Number* любое число или числовое выражение.

### Примеры использования

Exp(1) возвращает 2,71828...

## Int()

Возвращает целое число, не превышающее значение аргумента.

### Категория

Math (Математические).

### Синтаксис

```
IntegerResult = Int(Number);
```

### Аргументы

*Number* любое число или числовое выражение.

### Примечание

Если аргумент будет представлять собой отрицательное вещественное число, функция возвратит отрицательное целое число, которое по модулю больше значения аргумента.

### Примеры использования

```
Int(4.7); 'возвратит 4.7
```

```
Int(-4.7); 'возвратит -5
```

## IsBad()

Проверяет, является ли качество указанного атрибута плохим (Bad).

### Категория

Miscellaneous (Разные).

### Синтаксис

```
BooleanResult = IsBad(Attribute1, Attribute2, ...);
```

### Аргументы

*Attribute1*,  
*Attribute2*,  
... имена атрибутов, качество которых определяется. Их количество в аргументах функции может быть произвольным.

### Возвращаемые значения

Если индикатор качества хотя бы одного из указанных атрибутов равен "Bad", возвращается "True", в противном случае возвращается "False".

### Примеры использования

```
IsBad(TIC101.PV);
```

```
IsBad (TIC101.PV, PIC103.PV);
```

**См. также**

```
IsGood(), IsInitializing(), IsUncertain(), IsUsable()
```

## IsGood()

Проверяет, является ли качество указанного атрибута годным (Good).

**Категория**

Miscellaneous (Разные).

**Синтаксис**

```
BooleanResult = IsGood(Attribute1, Attribute2, ...);
```

**Аргументы**

<i>Attribute1,</i>	имена атрибутов, качество которых определяется. Их
<i>Attribute2,</i>	количество в аргументах функции может быть
...	произвольным.

**Возвращаемые значения**

Если индикатор качества хотя бы одного из указанных атрибутов равен "Good", возвращается "True", в противном случае возвращается "False".

**Примеры использования**

```
IsGood (TIC101.PV);
```

```
IsGood (TIC101.PV, PIC103.PV);
```

**См. также**

```
IsBad(), IsInitializing(), IsUncertain(), IsUsable()
```

## IsInitializing()

Проверяет, находится ли указанный атрибут в состоянии инициализации (индикатор качества = "Initializing").

**Категория**

Miscellaneous (Разные).

**Синтаксис**

```
BooleanResult = IsInitializing(Attribute1,  
Attribute2, ...);
```

**Аргументы**

<i>Attribute1,</i>	имена атрибутов, качество которых определяется. Их
<i>Attribute2,</i>	количество в аргументах функции может быть
...	произвольным.

**Возвращаемые значения**

Если индикатор качества хотя бы одного из указанных атрибутов равен "Initializing", возвращается "True", в противном случае возвращается "False".

**Примеры использования**

```
IsGood (TIC101.PV);
```

```
IsGood (TIC101.PV, PIC103.PV);
```

**См. также**

```
IsBad(), IsGood(), IsUncertain(), IsUsable()
```

## IsUncertain()

Проверяет, является ли качество указанного атрибута неопределённым (Uncertain).

### Категория

Miscellaneous (Разные).

### Синтаксис

```
BooleanResult = IsUncertain(Attribute1,  
Attribute2, ...);
```

### Аргументы

*Attribute1*, имена атрибутов, качество которых определяется. Их  
*Attribute2*, количество в аргументах функции может быть  
... произвольным.

### Возвращаемые значения

Если качество всех указанных атрибутов неопределённое (Uncertain), возвращается "True", в противном случае возвращается "False".

### Примеры использования

```
IsUncertain(TIC101.PV);  
IsUncertain(TIC101.PV, PIC103.PV);
```

### См. также

IsBad(), IsGood(), IsInitializing(), IsUsable()

## IsUsable()

Проверяет, можно ли использовать значения указанного атрибута в вычислениях.

### Категория

Miscellaneous (Разные).

### Синтаксис

```
BooleanResult = IsUsable(Attribute1, Attribute2, ...);
```

### Аргументы

*Attribute1*, имена атрибутов, проверяемых на предмет возможности  
*Attribute2*, использования в вычислениях. Их количество в  
... аргументах функции может быть произвольным.

### Возвращаемые значения

Если качество всех указанных атрибутов хорошее (Good) или неопределённое (Uncertain), возвращается "True", в противном случае возвращается "False".

### Примечание

Чтобы значение атрибута можно было использовать в вычислениях, нужно, чтобы оно было хорошего (Good) или неопределённого (Uncertain) качества. Кроме того, каждый атрибут вещественного типа (float) или типа с двойной точностью (double) не должен иметь значение NaN (нет).

### Примеры использования

```
IsUsable(TIC101.PV);  
IsUsable(TIC101.PV, PIC103.PV);
```

### См. также



IsBad(), IsGood(), IsInitializing(), IsUncertain()

## Log()

Возвращает значение натурального логарифма (по основанию  $e$ ) указанного аргумента.

### Категория

Math (Математические).

### Синтаксис

*RealResult* = Log(*Number*);

### Аргументы

*Number* любое число или числовое выражение.

### Примечание

Натуральный логарифм аргумента, равного нулю, не определён.

### Примеры использования

Log(100) ..... 4,605...

Log(1) ..... 0

### См. также

LogN(), Log10()

## LogN()

Возвращает значение логарифма указанного аргумента по указанному основанию.

### Категория

Math (Математические).

### Синтаксис

*RealResult* = LogN(*Number*, *Base*);

### Аргументы

*Number* любое число или числовое выражение.

*Base* целое число, задающее основание логарифма (может быть указан атрибут целого типа)

### Примечание

Логарифм аргумента по основанию 1 не определён.

### Примеры использования

LogN(8, 3) ..... 1,89279...

LogN(3, 7) ..... 0,564...

### См. также

Log(), Log10()

## Log10()

Возвращает значение десятичного логарифма (по основанию 10) указанного аргумента.

### Категория

Math (Математические).

**Синтаксис**

```
RealResult = Log10(Number);
```

**Аргументы**

*Number* любое число или числовое выражение.

**Примечание**

Натуральный логарифм аргумента, равного нулю, не определён.

**Примеры использования**

```
Log10(100) ..... 2
```

**См. также**

Log(), LogN()

**LogMessage()**

Передаёт пользовательское сообщение регистратору Logger.

Возвращает значение натурального логарифма (по основанию *e*) указанного аргумента.

**Категория**

Miscellaneous (Разные).

**Синтаксис**

```
LogMessage(msg);
```

**Аргументы**

*msg* сообщение, передаваемое регистратору Logger (литерал или атрибут символического типа).

**Примечание**

Это очень мощная функция, которая может быть использована для отладки скриптов. Указание вызовов функции в важных местах скрипта позволяет отслеживать порядок его исполнения, определять значения атрибутов до и после выполнения операторов QuickScript и т.д.

Каждое передаваемое регистратору Logger сообщение сопровождается точным значением даты и времени суток. Сообщение всегда начинается с элемента вида "TagName.ScriptName", что позволяет определить, какой объект и какой скрипт объекта передал данное сообщение.

**Примеры использования**

```
LogMessage("Report Script is Running");
```

При выполнении этого оператора в журнале появится строка следующего вида:

```
94/01/14 15:21:14 ScriptRuntime Message:Report Script is Running.
```

Другой пример:

```
LogMessage("The Value of MyTag is " + Text(MyTag, "#"));
```

```
MyTag = MyTag + 10;
```

```
LogMessage("The Value of MyTag is " + Text(MyTag, "#"));
```

**Now()**

Возвращает текущее время.

**Категория**

System (Системные).

**Синтаксис**

```
TimeValue = Now();
```

**Примечание**

Возвращаемое значение может быть отформатировано с помощью функций .NET или функции **StringFromTime()**.

**Pi()**

Возвращает значение числа π.

**Категория**

Math (Математические).

**Синтаксис**

```
RealResult = Pi();
```

**Примеры использования**

Pi() возвращает число 3,14159...

**Round()**

Возвращает результат округления вещественного числа с указанной точностью.

**Категория**

Math (Математические).

**Синтаксис**

```
RealResult = Round(Number, Precision);
```

**Аргументы**

*Number* любое число или выражение.

*Precision* точность округления (может представлять собой число или выражение).

**Примеры использования**

Round(4.3, 1): возвращает 4

Round(4.3, .01): возвращает 4.30

Round(4.5, 1): возвращает 5

Round(-4.5, 1): возвращает -4

Round(106, 5): возвращает 105

Round(43.7, 1): возвращает 43.5

**См. также**

Trunc()

**SendKeys()**

Передаёт приложению последовательность кодов клавиш. Принимающее приложение обрабатывает их так, словно они были получены из клавиатуры. С помощью этой функции можно имитировать ручной ввод данных или команд в приложение. В аргументах функции SendKeys() может быть указано большинство клавиатурных клавиш. Каждая из них представляется в виде одного или нескольких символов (например "A" для

обозначения клавиши с символом "A" или {ENTER} для обозначения клавиши ввода).

### Категория

Miscellaneous (Разные).

### Синтаксис

`SendKeys (KeySequence) ;`

### Аргументы

*KeySequence* последовательность обозначений клавиш или символьная переменная.

### Примечание

Чтобы задать более одной клавиши, их обозначения нужно объединить в одну строку. Например, для передачи кодов клавиш с символами доллара (\$) и буквы "b" нужно записать "\$b". В следующей таблице приведены обозначения клавиш со специальными функциями:

Клавиша	Обозначение	Клавиша	Обозначение
BACKSPACE	{BACKSPACE} или {BS}	HOME	{HOME}
BREAK	{BREAK}	INSERT	{INSERT}
CAPLOCKS	{CAPLOCKS}	LEFT	{LEFT}
DELETE	{DELETE} или {DEL}	NUMLOCK	{NUMLOCK}
DOWN	{DOWN}	PAGE DOWN	{PGDN}
END	{END}	PAGE UP	{PGUP}
ENTER	{ENTER} или ~ (тильда)	PRTSC	{PRTSC}
ESCAPE	{ESCAPE} или {ESC}	RIGHT	{RIGHT}
F1	{F1}*	TAB	{TAB}
		UP	{UP}

\* все остальные функциональные клавиши обозначаются аналогичным образом.

Специальные клавиши Shift, Alt и Ctrl обозначаются по-иному:

Клавиша	Обозначение
SHIFT	+ (знак плюса)
CTRL	^ (знак вставки)
ALT	% (знак процента)

На некоторых компьютерах эта функция может быть неприменимой из-за особенностей реализации уровня абстрагирования оборудования HAL в операционной системе Windows.

### Примеры использования

Если рядом должны быть указаны две специальные клавиши, нужно использовать скобки. Следующий вызов функции имитирует нажатие клавиши CTRL, затем клавиши ALT, затем клавиши с буквой "p":

`SendKeys ("^(%p)");`

Перед вызовом этой функции может быть указан оператор активизации приложения, которому будет передана указанная последовательность кодов клавиш.

Следующие операторы активизируют приложение Excel и передают ему коды комбинации клавиш CTRL+P (запуск макроса печати электронной таблицы на принтере):

```
ActivateApp ("MS Excel " ) ;  
SendKeys ( " ^ ( p ) " ) ;
```

## SetBad()

Устанавливает индикатор качества указанного атрибута в "Bad".

### Категория

Miscellaneous (Разные).

### Синтаксис

```
SetBad (Attribute) ;
```

### Аргументы

*Attribute* имя атрибута, для которого индикатор качества устанавливается в "Bad".

### Примечание

Указанный атрибут должен быть одним из атрибутов объекта, с которым связан исполняющийся скрипт.

### Примеры использования

```
SetBad (me . PV) ;
```

### См. также

SetGood(), SetInitializing(), SetUncertain()

## SetGood()

Устанавливает индикатор качества указанного атрибута в "Good".

### Категория

Miscellaneous (Разные).

### Синтаксис

```
SetGood (Attribute) ;
```

### Аргументы

*Attribute* имя атрибута, для которого индикатор качества устанавливается в "Good".

### Примечание

Указанный атрибут должен быть одним из атрибутов объекта, с которым связан исполняющийся скрипт.

### Примеры использования

```
SetGood (me . PV) ;
```

### См. также

SetBad(), SetInitializing(), SetUncertain()

## SetInitializing()

Устанавливает индикатор качества указанного атрибута в "Good".

### Категория

Miscellaneous (Разные).

### Синтаксис

```
SetInitializing(Attribute);
```

### Аргументы

*Attribute* имя атрибута, для которого индикатор качества устанавливается в "Initializing".

### Примечание

Указанный атрибут должен быть одним из атрибутов объекта, с которым связан исполняющийся скрипт.

### Примеры использования

```
SetInitializing(me.PV);
```

### См. также

SetBad(), SetGood(), SetUncertain()

## SetUncertain()

Устанавливает индикатор качества указанного атрибута в "Uncertain".

### Категория

Miscellaneous (Разные).

### Синтаксис

```
SetUncertain(Attribute);
```

### Аргументы

*Attribute* имя атрибута, для которого индикатор качества устанавливается в "Uncertain".

### Примечание

Указанный атрибут должен быть одним из атрибутов объекта, с которым связан исполняющийся скрипт.

### Примеры использования

```
SetUncertain(me.PV);
```

### См. также

SetBad(), SetGood(), SetInitializing()

## Sgn()

Возвращает знак числового значения (положительного, отрицательного или нулевого).

### Категория

Math (Математические).

### Синтаксис

```
IntegerResult = Sgn(Number);
```

### Аргументы

*Number* любое число или числовое выражение.

### Примечание

Если аргумент положительный, функция возвратит 1, если отрицательный – -1, если нулевой – возвратит 0.

### Примеры использования

`Sgn(425)` возвратит 1

`Sgn(0)` возвратит 0

`Sgn(-37.3)` возвратит -1

## Sin()

Возвращает значение синуса угла, выраженного в градусах.

### Категория

Math (Математические).

### Синтаксис

`Result = Sin(Number);`

### Аргументы

*Number* значение угла в градусах (число или числовое выражение).

### Примеры использования

`Sin(90)` возвращает 1

`Sin(0)` возвращает 0

Ниже показан пример использования функции в математическом выражении:

`Wave = 100 * Sin(6 * Now().Second);`

### См. также

`Cos()`, `Tan()`, `ArcCos()`, `ArcSin()`, `ArcTan()`

## Sqrt()

Возвращает значение квадратного корня из числа.

### Категория

Math (Математические).

### Синтаксис

`RealResult = Sqrt(Number);`

### Аргументы

*Number* любое число или числовое выражение.

### Примеры использования

`x = Sqrt(me.PV);`

## StringASCII()

Возвращает код ASCII первого символа указанной строки.

### Категория

String (Символьные).

### Синтаксис

```
IntegerResult = StringASCII(Char);
```

### Аргументы

*Char* алфавитноцифровой символ, строка или символьный атрибут.

### Примечание

При исполнении данной функции определяется код единственного символа. Если в качестве аргумента указана строка символов, функция определяет код ASCII только первого символа.

### Примеры использования

```
StringASCII("A") возвращает 65
```

```
StringASCII("A Mixer Is Running") возвращает тоже 65
```

```
StringASCII("a Mixer Is Running") возвращает 97
```

### См. также

StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringChar()

Возвращает символ, код ASCII которого равен указанному значению.

### Категория

String (Символьные).

### Синтаксис

```
StringResult = StringChar(ASCII);
```

### Аргументы

*ASCII* код ASCII или атрибут целого типа.

### Примечание

С помощью этой функции можно, например, задавать для атрибута символы, не представленные на клавиатуре компьютера.

### Примеры использования

В следующем примере эта функция используется для добавления в конец строки StringAttribute символов возврата каретки (код ASCII 13) и перевода строки (код ASCII 10) и присваиванию результата переменной ControlString. Данная функция позволяет вставлять в текст, отображаемый на экране или выводимый на принтер, символы, управляющие видом текста или работой устройства вывода, такого как принтер или модем.

```
ControlString = StringAttribute _ StringChar(13) +  
StringChar(10);
```

Эта функция может также использоваться в SQL-операторах для указания в символьных строках конструкции WHERE символов двойных кавычек (код ASCII 34).

### См. также

StringASCII(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()



## StringFromIntg()

Возвращает символьное представление целого числа в указанной системе счисления.

### Категория

String (Символьные).

### Синтаксис

```
StringResult = StringFromIntg(Number, Base);
```

### Аргументы

*Number* любое число или числовое выражение.  
*Base* основание системы счисления, используемое для определения символьного представления указанного числа в этой системе.

### Примеры использования

StringFromIntg(26, 2) возвращает "11010"

StringFromIntg(26, 8) возвращает "32"

StringFromIntg(26, 16) возвращает "1A"

### См. также

StringASCII(), StringChar(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringFromReal()

Возвращает символьное представление вещественного числа (как числа с плавающей запятой или в экспоненциальном представлении).

### Категория

String (Символьные).

### Синтаксис

```
StringResult = StringFromReal(Number, Precision, Type);
```

### Аргументы

*Number* преобразуемое число или числовое выражение.  
*Precision* количество позиций в дробной части результата (целое число или атрибут целого типа).  
*Type* символьный аргумент, задающий вид представления:  
"f" в виде числа с плавающей запятой;  
"e" в экспоненциальном виде с символом показателя степени в нижнем регистре ("e");  
"E" в экспоненциальном виде с символом показателя степени в верхнем регистре ("E");

### Примеры использования

StringFromReal(263.355, 2, "f") возвращает "263.36"

StringFromReal(263.355, 2, "e") возвращает "2.63e2"

StringFromReal(263.355, 3, "E") возвращает "2.634E2"

### См. также

StringASCII(), StringChar(), StringFromIntg(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringFromTime()

Преобразует число секунд, прошедших с 1 января 1970 г., в символьное представление в указанном формате.

### Категория

String (Символьные).

### Синтаксис

```
StringResult = StringFromTime(SecsSince1-1-70,
StringType);
```

### Аргументы

*SecsSince1-1-70* преобразуемое значение,

*StringType* вид результата:

- |   |  |
|---|--|
| 1 | формат даты определяется установками в Панели управления Windows,  |
| 2 | формат показаний времени определяется установками в Панели управления Windows,                               |
| 3 | результат длиной 24 символа содержит как дату, так и показания времени, например: "Wed Jan 02 02:03:55 1993" |
| 4 | результат представляет собой укороченное обозначение дня недели: "Wed"                                       |
| 5 | результат представляет собой полное обозначение дня недели: "Wednesday"                                      |

### Примечание

Преобразуемое значение должно соответствовать показаниям времени в формате UTC (Universal Time Coordinated – Единое координированное время, представляющее число секунд, прошедших с 1 января 1970 г. по гринвичскому времени). Возвращаемое значение указывается по местному времени

### Примеры использования

StringFromTime(86400, 1) возвращает "1/2/70"

StringFromTime(86400, 2) возвращает "12:00:00 AM"

StringFromTime(86400, 3) возвращает "Fri Jan 02 00:00:00 1970"

StringFromTime(86400, 4) возвращает "Fri"

StringFromTime(86400, 5) возвращает "Friday"

### См. также

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringInString()

Возвращает номер позиции в строке, в которой первый раз встречается указанная подстрока.

## Категория

String (Символьные).

## Синтаксис

```
IntegerResult = StringInString(Text, SearchFor,  
    StartPos, CaseSens);
```

## Аргументы

<i>Text</i>	строка, в которой должен осуществляться поиск подстроки (литерал или имя символьного атрибута);
<i>SearchFor</i>	искомая подстрока (литерал или имя символьного атрибута);
<i>StartPos</i>	номер начальной позиции в строке, начиная с которой нужно осуществлять поиск вхождения указанной подстроки;
<i>CaseSens</i>	аргумент, определяющий необходимость учёта регистра символов: 0 = не учитывать, 1 = учитывать регистр. Может быть задан в виде числа или в виде переменной целого типа.

## Примечание

Если в исходной строке присутствует несколько указанных подстрок, возвращаемый номер позиции будет соответствовать первому найденному вхождению подстроки.

## Примеры использования

```
StringInString("The mixer is running", "mix", 1, 0)  
возвращает 5
```

```
StringInString("Today is Thursday", "day", 1, 0)  
возвращает 3
```

```
StringInString("Today is Thursday", "day", 10, 0)  
возвращает 15
```

```
StringInString("Today is Veteran's Day", "Day", 1, 1)  
возвращает 20
```

```
StringInString("Today is Veteran's Day", "Night", 1,  
1) возвращает 0
```

## См. также

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringLeft()

Возвращает указанное количество символов строки, начиная с первого.

## Категория

String (Символьные).

## Синтаксис

```
StringResult = StringLeft(Text, Chars);
```

## Аргументы

<i>Text</i>	исходная строка (литерал или имя символьного атрибута);
<i>Chars</i>	количество возвращаемых символов (число или переменная целого типа).

**Примечание**

Если значение аргумента `Chars` равно 0, возвращается вся исходная строка.

**Примеры использования**

```
StringLeft("The Control Pump is On", 3); 'возвращает "The"
```

```
StringLeft("Pump 01 is On", 4); 'возвращает "Pump"
```

```
StringLeft("Pump 01 is On", 96); 'возвращает "Pump 01 is On"
```

```
StringLeft("The Control Pump is On", 0); 'возвращает "The  
Control Pump is On"
```

**См. также**

`StringASCII()`, `StringChar()`, `StringFromIntg()`, `StringFromReal()`,  
`StringFromTime()`, `StringInString()`, `StringLen()`, `StringLower()`, `StringMid()`,  
`StringReplace()`, `StringRight()`, `StringSpace()`, `StringTest()`, `StringToIntg()`,  
`StringToReal()`, `StringTrim()`, `StringUpper()`, `Text()`

## StringLen()

Возвращает количество всех символов в указанной строке.

**Категория**

String (Символьные).

**Синтаксис**

```
IntegerResult = StringLen(Text);
```

**Аргументы**

*Text* исходная строка (литерал или имя символьного атрибута).

**Примечание**

Функция подсчитывает количество всех символов в строке, включая управляющие символы и символы, которые обычно не отображаются на экране монитора.

**Примеры использования**

```
StringLen("Twelve percent") возвращает 14
```

```
StringLen("12%") возвращает 3
```

```
StringLen("The end." + StringChar(13)); 'возвращает 9
```

13 представляет собой код ASCII символа возврата каретки.

**См. также**

`StringASCII()`, `StringChar()`, `StringFromIntg()`, `StringFromReal()`,  
`StringFromTime()`, `StringInString()`, `StringLeft()`, `StringLower()`, `StringMid()`,  
`StringReplace()`, `StringRight()`, `StringSpace()`, `StringTest()`, `StringToIntg()`,  
`StringToReal()`, `StringTrim()`, `StringUpper()`, `Text()`

## StringLower()

Преобразует все прописные буквы указанной строки в строчные (то есть преобразует из верхнего регистра в нижний).

**Категория**

String (Символьные).

**Синтаксис**

```
StringResult = StringLower(Text);
```

**Аргументы**

*Text* исходная строка (литерал или имя символического атрибута).

#### Примечание

Буквы в нижнем регистре, графические знаки, цифры и прочие специальные символы не преобразуются.

#### Примеры использования

`StringLower("TURBINE")` возвращает "turbine"

`StringLower("22.2 Is the Value");` возвращает "22.2 is the Value"

#### См. также

`StringASCII()`, `StringChar()`, `StringFromIntg()`, `StringFromReal()`, `StringFromTime()`, `StringInString()`, `StringLeft()`, `StringLen()`, `StringMid()`, `StringReplace()`, `StringRight()`, `StringSpace()`, `StringTest()`, `StringToIntg()`, `StringToReal()`, `StringTrim()`, `StringUpper()`, `Text()`

## StringMid()

Извлечение из строки указанного количества символов, начиная с заданной позиции.

#### Категория

String (Символьные).

#### Синтаксис

```
StringResult = StringMid(Text, StartChar, Chars);
```

#### Аргументы

*Text* исходная строка (литерал или имя символического атрибута);

*StartChar* номер позиции первого символа извлекаемой подстроки (любое число или переменная целого типа);

*Chars* количество извлекаемых символов (любое число или переменная целого типа).

#### Примечание

Данная функция отличается от функций `StringLeft()` и `StringRight()` тем, что она позволяет задавать как начало, так и конец извлекаемой подстроки.

#### Примеры использования

`StringMid("The Furnace is Overheating", 5, 7)`  
возвращает "Furnace"

`StringMid("The Furnace is Overheating", 13, 3)`  
возвращает "is "

`StringMid("The Furnace is Overheating", 16, 50)`  
возвращает "Overheating"

#### См. также

`StringASCII()`, `StringChar()`, `StringFromIntg()`, `StringFromReal()`, `StringFromTime()`, `StringInString()`, `StringLeft()`, `StringLen()`, `StringLower()`, `StringReplace()`, `StringRight()`, `StringSpace()`, `StringTest()`, `StringToIntg()`, `StringToReal()`, `StringTrim()`, `StringUpper()`, `Text()`

## StringReplace()

Модификация указанной части исходной строки.

#### Категория

String (Символьные).

### Синтаксис

```
StringResult = StringReplace(Text, SearchFor,
    ReplaceWith, CaseSens, NumToReplace,
    MatchWholeWords);
```

### Аргументы

<i>Text</i>	исходная модифицируемая строка (литерал или имя символьного атрибута);
<i>SearchFor</i>	искомая подстрока, которую требуется заменить новой подстрокой (литерал или имя символьного атрибута);
<i>ReplaceWith</i>	строка замены (литерал или имя символьного атрибута);
<i>CaseSens</i>	аргумент, определяющий необходимость учёта регистра символов при поиске заменяемой подстроки: 0 = не учитывать, 1 = учитывать регистр. Может быть задан в виде числа или в виде переменной целого типа.
<i>NumToReplace</i>	количество замен, которые нужно выполнять в исходной модифицируемой строке (число или переменная целого типа). Чтобы указать необходимость замены всех вхождений заменяемой подстроки, этому аргументу нужно присвоить значение -1.
<i>MatchWholeWords</i>	аргумент, определяющий необходимость замены только целых слов: 0 = нет, 1 = да (число или переменная целого типа). Если значение этого аргумента равно 1, а значение аргумента SearchFor равно "and", символы "and" в слове "handle" будут оставлены без изменений. Если же значение аргумента MatchWholeWords равно 0, это слово будет изменено.

### Примечание

С помощью этой функции в символьных атрибутах можно заменять отдельные символы, слова и целые предложения.

Функция StringReplace() не распознаёт специальные символы, такие как @#\$%&\*(), трактуя их как символы-разделители. Например, оператор вида StringReplace(abc#,abc#,1234,0,1,1) не будет выполнять никаких замен. Символ "#" будет обрабатываться как разделитель.

### Примеры использования

```
StringReplace("In From Within", "In", "Out", 0, 1, 0)
возвращает "Out From Within" (заменяется только первое вхождение
указанной подстроки)
```

```
StringReplace("In From Within", "In", "Out", 0, -1, 0)
возвращает "Out From Without" ()
```

```
StringReplace("In From Within", "In", "Out", 1, -1, 0)
возвращает "Out From Within" (заменяются все вхождения указанной
подстроки с учётом регистра)
```

```
StringReplace("In From Within", "In", "Out", 0, -1, 1)
возвращает "Out From Within" (заменяются все вхождения указанной
подстроки, представляющие собой отдельные слова)
```

**См. также**

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringRight()

Возвращает указанное количество символов строки, начиная с последнего.

### Категория

String (Символьные).

### Синтаксис

```
StringResult = StringRight(Text, Chars);
```

### Аргументы

*Text* исходная строка (литерал или имя символьного атрибута);  
*Chars* количество возвращаемых символов (число или переменная целого типа).

### Примечание

Если значение аргумента Chars равно 0, возвращается вся исходная строка.

### Примеры использования

```
StringRight("The Pump is On", 2); 'возвращает "On"  
StringRight("The Pump is On", 5); 'возвращает "is On"  
StringRight("The Pump is On", 87); 'возвращает "The Pump is On"  
StringRight("The Pump is On", 0); 'возвращает "The Pump is On"
```

### См. также

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringSpace()

Возвращает строку, состоящую из указанного количества пробелов.

### Категория

String (Символьные).

### Синтаксис

```
StringResult = StringSpace(NumSpaces);
```

### Аргументы

*NumSpaces* количество пробелов (число или переменная целого типа).

### Примеры использования

В следующих примерах символы пробела представлены знаком "x"

```
StringSpace(4) возвращает "xxxx"  
"Pump" + StringSpace(1) + "Station" возвращает  
"PumpxStation"  
StringRight("The Pump is On", 2) возвращает "On"  
StringRight("The Pump is On", 5) возвращает "is On"
```

`StringRight("The Pump is On", 87)` возвращает "The Pump is On"

`StringRight("The Pump is On", 0)` возвращает "The Pump is On"

#### См. также

`StringASCII()`, `StringChar()`, `StringFromIntg()`, `StringFromReal()`,  
`StringFromTime()`, `StringInString()`, `StringLeft()`, `StringLen()`, `StringLower()`,  
`StringMid()`, `StringReplace()`, `StringRight()`, `StringTest()`, `StringToIntg()`,  
`StringToReal()`, `StringTrim()`, `StringUpper()`, `Text()`

## StringTest()

Проверка типа первого символа указанной строки.

#### Категория

String (Символьные).

#### Синтаксис

```
DiscreteResult = StringTest(Text, TestType);
```

#### Аргументы

*Text* исходная строка (литерал или атрибут символьного типа).

*TestType* индикатор типа. Возможные значения:

- |    |   |
|----|---|
| 1  | алфавитноцифровые символы ('A'-'Z', 'a'-'z' и '0'-'9')      |
| 2  | цифры ('0'-'9')   |
| 3  | буквы ('A'-'Z', 'a'-'z')                                    |
| 4  | буквы в верхнем регистре ('A'-'Z')                          |
| 5  | буквы в нижнем регистре ('a'-'z')                           |
| 6  | знаки препинания (0x21-0x2F)                                |
| 7  | символы ASCII (0x00-0x7F)                                   |
| 8  | шестнадцатеричные цифры ('A'-'F', или 'a'-'f', или '0'-'9') |
| 9  | печатные символы ('0x20-0x7E)                               |
| 10 | управляющие символы (0x00-0x1F и 0x7F)                      |
| 11 | символы пропуска (0x09-0x0D и 0x20)                         |

#### Примечание

Функция возвращает "True", если тип первого символа аргумента *Text* совпадает с типом, заданным аргументом *TestType*, в противном случае возвращает "False". Если аргумент *Text* представляет собой многосимвольную строку, проверяется только первый символ.

#### Примеры использования

`StringTest("ABC123", 1)` возвращает 1

`StringTest("ABC123", 5)` возвращает 0

#### См. также

`StringASCII()`, `StringChar()`, `StringFromIntg()`, `StringFromReal()`,  
`StringFromTime()`, `StringInString()`, `StringLeft()`, `StringLen()`, `StringLower()`,  
`StringMid()`, `StringReplace()`, `StringRight()`, `StringSpace()`, `StringToIntg()`,  
`StringToReal()`, `StringTrim()`, `StringUpper()`, `Text()`

## StringToIntg()

Преобразование числа, представленного в виде строки символов, в значение целого типа.



### Категория

String (Символьные).

### Синтаксис

```
IntegerResult = StringToIntg(Text);
```

### Аргументы

*Text* преобразуемая строка символов (литерал или атрибут символьного типа).

### Примечание

Во время преобразования сначала анализируется значение первого символа. Если первый символ строки не является цифрой (символы пропуска, то есть пробелы и символы табуляции игнорируются), функция прекращает преобразование и возвращает 0. Если первый символ является цифрой, функция анализирует очередной символ строки, пока не достигнет конца строки или не встретит нецифровой символ.

### Примеры использования

StringToIntg("ABCD") возвращает 0

StringToIntg("22.2 is the Value") возвратит 22, поскольку преобразование закончится при обнаружении символа точки

StringToIntg("The value is 22") возвращает 0

### См. также

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToReal(), StringTrim(), StringUpper(), Text()

## StringToReal()

Преобразование числа, представленного в виде строки символов, в значение вещественного типа (с плавающей запятой).

### Категория

String (Символьные).

### Синтаксис

```
RealResult = StringToReal(Text);
```

### Аргументы

*Text* преобразуемая строка символов (литерал или атрибут символьного типа).

### Примечание

Во время преобразования сначала анализируется значение первого символа. Если первый символ строки не является цифрой (символы пропуска, то есть пробелы и символы табуляции игнорируются), функция прекращает преобразование и возвращает 0. Если первый символ является цифрой, функция анализирует очередной символ строки, пока не достигнет конца строки или не встретит нецифровой символ.

### Примеры использования

StringToReal("ABCD") возвращает 0

StringToIntg("22.261 is the Value") возвратит 22.261

StringToIntg("The value is 2") возвращает 0

**См. также**

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(),  
StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(),  
StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(),  
StringToIntg(), StringTrim(), StringUpper(), Text()

**StringTrim()**

Удаление пробелов из указанной строки символов.

**Категория**

String (Символьные).

**Синтаксис**

```
StringResult = StringTrim(Text, TrimType);
```

**Аргументы**

<i>Text</i>	исходная строка символов (литерал или атрибут символьного типа).
<i>TrimType</i>	индикатор удаляемых пробелов: <ol style="list-style-type: none"> <li>1 начальные пробелы (с начала строки до первого символа, отличного от пробела)</li> <li>2 конечные пробелы (после последнего символа, отличного от пробела, до конца строки)</li> <li>3 все пробелы за исключением одиночных пробелов между словами</li> </ol>

**Примечание**

Функция удаляет из строки символы пропуска (с кодами ASCII 0x009-0x0D и 0x020), расположение которых задаётся аргументом TrimType.

**Примеры использования**

В следующих примерах символы пропуска обозначены знаком "x".

```
StringTrim("xxxxxThisxisxaxxtestxxxxx", 1) возвратит  
строку "Thisxisaxxtestxxxxx"
```

```
StringTrim("xxxxxThisxisxaxxtestxxxxx", 2) возвратит  
строку "xxxxxThisxisaxxtest"
```

```
StringTrim("xxxxxThisxisxaxxtestxxxxx", 3) возвратит  
строку "Thisxisaxtest"
```

Удалить из строки все пробелы можно с помощью функции

**StringReplace()**, указав символ "null" в её аргументах как строку замены.

**См. также**

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(),  
StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(),  
StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(),  
StringToIntg(), StringToReal(), StringUpper(), Text()

**StringUpper()**

Преобразует все строчные буквы указанной строки в прописные (то есть преобразует из нижнего регистра в верхний).

**Категория**

String (Символьные).

**Синтаксис**

```
StringResult = StringUpper(Text);
```

### Аргументы

*Text* исходная строка (литерал или имя символического атрибута);

### Примечание

Буквы в верхнем регистре, графические знаки, цифры и прочие специальные символы не преобразуются.

### Примеры использования

```
StringLower("abcd") возвращает "ABCD"
```

```
StringLower("22.2 is the Value"); 'возвращает "22.2 IS THE VALUE"
```

### См. также

StringASCII(), StringChar(), StringFromIntg(), StringFromReal(), StringFromTime(), StringInString(), StringLeft(), StringLen(), StringLower(), StringMid(), StringReplace(), StringRight(), StringSpace(), StringTest(), StringToIntg(), StringToReal(), StringTrim(), Text()

## Tan()

Возвращает значение тангенса угла, указанного в градусах.

### Категория

Math (Математические).

### Синтаксис

```
Result = Tan(Number)
```

### Аргументы

*Number* угол в градусах (любое число или числовое выражение).

### Примеры использования

Tan(45) возвращает 1.

Tan(0) возвращает 0.

В следующем примере показано, как эта функция используется в выражениях:

```
Wave = 10 + 50 * Tan(6 * Now().Second);
```

### См. также

Cos(), Sin(), ArcCos(), ArcSin(), ArcTan()

## Text()

Преобразование числа в символическое представление в соответствии с заданным форматом.

### Категория

String (Символьные).

### Синтаксис

```
StringResult = Text(Number, Format);
```

### Аргументы

*Number* любое число или атрибут числового типа.

*Format* формат преобразования (литерал или атрибут символического типа).

### Примеры использования

`Text (66, "#.00")` возвращает строку "66.00"

`Text (22.268, "#.00")` возвращает строку "22.27"

`Text (9.999, "#.00")` возвращает строку "10.00"

В следующем примере показано, как эта функция используется в аргументах других функций:

```
LogMessage("The current value of FreezerRoomTemp is: " + Text (FreezerRoomTemp, "#.#"));
```

В следующем примере переменная `MessageTag` получит значение "One=1 Two=2":

```
MessageTag = "One=" + Text (1, "#") + StringChar (32) + "Two=" + Text (2, "#");
```

### См. также

`StringFromIntg()`, `StringToIntg()`, `StringFromReal()`, `StringToReal()`

## Trunc()

Округление вещественного числа (с плавающей запятой) путём отбрасывания дробной части.

### Категория

Math (Математические).

### Синтаксис

```
NumericResult = Trunc (Number)
```

### Аргументы

*Number* любое число или числовое выражение.

### Примечание

Функция возвращает такой же результат, как и оператор присваивания атрибуту целого типа – значения атрибута вещественного типа.

### Примеры использования

`Trunc (4.3)` возвращает 4

`Trunc (-4.3)` возвращает -4

### См. также

`Round()`

## WriteStatus()

Возвращает статус выполнения последней операции записи в указанный атрибут.

### Категория

Miscellaneous (Разные).

### Синтаксис

```
Result = WriteStatus (Attribute);
```

### Аргументы

*Attribute* имя атрибута.

### Возвращаемые значения

Статус записи в атрибут может быть одним из следующих:

- MxStatusOk
- MxStatusPending
- MxStatusWarning
- MxStatusCommunicationError
- MxStatusConfigurationError
- MxStatusOperationalError
- MxStatusSecurityError
- MxStatusSoftwareError
- MxStatusOtherError

#### Примечание

Если в указанный атрибут запись значений ещё не производилась, функция возвратит значение MxStatusOk. Это же значение всегда возвращается при определении статуса записи в атрибуты, для которых вычисляемое качество (отличное от "Good") не поддерживается.

#### Примеры использования

```
WriteStatus (TIC101.PV);
```

## WWControl()

Восстановление размеров, минимизация, максимизация или закрытие окна приложения.

#### Категория

Miscellaneous (Разные).

#### Синтаксис

```
WWControl (AppTitle, ControlType);
```

#### Аргументы

<i>AppTitle</i>	название приложения, с окном которого требуется выполнить указанное действие (литерал или атрибут символьного типа).
<i>ControlType</i>	аргумент, определяющий действие, которое должно быть выполнено. Допустимые значения приведены ниже. Указанные действия аналогичны действиям, выполняемым при щелчке кнопкой мыши соответствующих пунктов управляющего меню. В качестве аргумента может быть указан литерал или атрибут символьного типа.
"Restore"	активизация приложения и показ его окна;
"Minimize"	активизация окна и показ его в виде значка;
"Maximize"	активизация приложения и показ его окна;
"Close"	завершение работы приложения.

#### Примеры использования

```
WWControl ("Calculator", "Restore");
```

#### См. также

```
ActivateApp()
```

## WVExecute()

Передача команды в указанное приложение (по протоколу DDE).

### Категория

WVDDDE

### Синтаксис

```
Status = WVExecute(Application, Topic, Command);
```

### Аргументы

<i>Application</i>	название приложения, которое должно исполнить полученную команду (литерал или атрибут символьного типа);
<i>Topic</i>	обозначение группы данных внутри приложения (литерал или атрибут символьного типа);
<i>Command</i>	передаваемая команда (литерал или атрибут символьного типа).

### Возвращаемые значения

*Status* является переменной целого типа, получающей значения 1, -1 или 0. Функция WVExecute() возвращает 1, если указанное приложение исполняется, указанная группа данных существует и передача команды была выполнена успешно. 0 возвращается, если приложение занято, и -1, – если при выполнении функции возникла ошибка.

### Примечание

Значение аргумента *Command* будет передано указанной группе данных в указанном приложении.

---

**Внимание!** При использовании функции WVExecute() в синхронных скриптах нужно соблюдать следующие условия:

1. Не запускать её циклически (то есть вызывать её снова и снова),
2. Не указывать несколько вызовов функции подряд в одном и том же скрипте.
3. Не пользоваться ею для выполнения занимающих много времени задач в другом DDE-приложении.

Вместе с тем эти условия можно не соблюдать при использовании функции в асинхронных скриптах.

---

### Примеры использования

Следующий скрипт запускает макрос TestMacro в приложении Excel.

```
Macro = "Macro1!TestMacro";  
Command = "[Run(" + StringChar(34) + Macro +  
StringChar(34) + ", 0)]";  
WVExecute("Excel", "system", Command);
```

При выполнении этого оператора в приложение Excel будет передана следующая команда (и запущен макрос TestMacro):

```
[Run("Macro1!TestMacro")]
```

Ниже приведён оператор передачи команды MyMacro в приложение Microsoft Access:

```
WVExecute("MSAccess", "system", "MyMacro");
```

## WWPoke()

Запись значения в указанный элемент данных в группе данных указанного приложения (по протоколу DDE).

### Категория

WWDDE

### Синтаксис

```
Status = WWPoke(Application, Topic, Item, TextValue);
```

### Аргументы

<i>Application</i>	имя приложения, в элемент данных которого нужно записать указанное значение (литерал или атрибут символьного типа);
<i>Topic</i>	обозначение группы данных внутри приложения (литерал или атрибут символьного типа);
<i>Item</i>	имя элемента данных в группе, в который нужно записать указанное значение (литерал или атрибут символьного типа);
<i>TextValue</i>	передаваемое значение. Если оно является числом, его нужно преобразовать в символьный вид, например с помощью функций <b>Text()</b> , <b>StringFromIntg()</b> или <b>StringFromReal()</b> (литерал или атрибут символьного типа).

### Возвращаемые значения

*Status* является переменной целого типа, получающая значения 1, -1 или 0. Функция WWPoke() возвращает 1, если указанное приложение выполняется, указанные группа и элемент данных существуют и передача команды была выполнена успешно. 0 возвращается, если приложение занято, и -1 – если при выполнении функции возникла ошибка.

### Примечание

Значение аргумента *TextValue* будет передано в указанный элемент группы данных указанного приложения.

---

**Внимание!** При использовании функции WWPoke() в синхронных скриптах нужно соблюдать следующие условия:

1. Не запускать её циклически (то есть вызывать её снова и снова),
2. Не указывать несколько вызовов функции подряд в одном и том же скрипте.
3. Не пользоваться ею для выполнения занимающих много времени задач в другом DDE-приложении.

Вместе с тем эти условия можно не соблюдать при использовании функции в асинхронных скриптах.

---

### Примеры использования

Следующий скрипт записывает значение переменной *Value* в указанную ячейку электронной таблицы Excel.

```
String = Text(Value, "0");  
WWPoke("Excel", "[Book1.xls]sheet1", "r1c1", String);
```

Поведение функции WWPoke() при передаче данных из приложения View в приложение View не определено, и такой вызов не поддерживается. В этом случае успешное завершение команды не гарантируется. Вероятнее всего, что её исполнение будет прервано без достижения желаемых результатов.

**См. также**

Text(), StringFromIntg(), StringFromReal()

**WWRequest()**

Данная функция выдаёт разовый запрос на получение значения указанного элемента группы данных в указанном приложении.

**Категория**

WWDDE

**Синтаксис**

```
Status = WWRequest(Application, Topic, Item,
Attribute);
```

**Аргументы**

<i>Application</i>	имя приложения, из которого требуется получить значение указанного элемента данных (литерал или атрибут символьного типа);
<i>Topic</i>	обозначение группы данных внутри приложения (литерал или атрибут символьного типа);
<i>Item</i>	имя элемента данных в группе, в который нужно записать указанное значение (литерал или атрибут символьного типа);
<i>Attribute</i>	имя атрибута символьного типа, заключённое в скобки, в который будет записано полученное из указанного приложения значение (литерал или имя атрибута символьного типа).

**Возвращаемые значения**

*Status* является переменной целого типа, получающая значения 1, -1 или 0. Функция WWRequest() возвращает 1, если указанное приложение исполняется, указанные группа и элемент данных существуют и передача значения была выполнена успешно. 0 возвращается, если приложение занято, и -1, – если при выполнении функции возникла ошибка.

**Примечание**

В указанный атрибут по протоколу DDE будет передано значение указанного элемента группы данных указанного приложения.

Значение возвращается в символьной форме. Преобразовать его в числовое значение можно с помощью функций **StringToIntg()** и **StringToReal()**.

---

**Внимание!** При использовании функции WWRequest() в синхронных скриптах нужно соблюдать следующие условия:

1. Не запускать её циклически (то есть вызывать её снова и снова),
2. Не указывать несколько вызовов функции подряд в одном и том же скрипте.
3. Не пользоваться ею для выполнения занимающих много времени задач в другом DDE-приложении.

Вместе с тем эти условия можно не соблюдать при использовании функции в асинхронных скриптах.

---

**Примеры использования**

Следующий скрипт считывает значение, хранящееся в указанной ячейке электронной таблицы Excel, и преобразует её в вещественное число.

```
WWRequest("Excel", "[Book1.xls]sheet1", "r1c1",
Result);
```

---



```
Value = StringToReal(Result);
```

**См. также**

```
StringToIntg(), StringToReal()
```

## Переменные QuickScript .NET

Прежде чем использовать в скриптах переменные, их нужно объявить. Объявленные переменные могут применяться как в левой, так и в правой части операторов и выражений.

Локальные переменные могут использоваться в тех же случаях, что и атрибуты объектов. Однако при достижении конца функции локальные переменные теряют своё значение (кроме тех переменных, которые объявлены в общей секции скрипта). Доступ к переменным, объявленным в общей секции скрипта, из других скриптов невозможен. Объявленные переменные сохраняют своё значение в течение всего периода существования объекта, с которым связан соответствующий скрипт.

Каждая переменная должна быть объявлена отдельным оператором DIM, завершающимся точкой с запятой. Синтаксис оператора DIM следующий:

```
DIM <VariableName> [(UpperBound [, UpperBound [,  
UpperBound ]]) ] [AS DataType ];
```

где

DIM	требуемое ключевое слово;
VariableName	имя переменной, начинающееся с буквы ('A'-'Z' или 'a'-'z'). При этом остальные символы имени могут быть буквами, цифрами и символом подчёркивания ("_"). Длина имени не может превышать 255 символов в кодировке Unicode;
UpperBound	число от 1 до 2147483647, обозначающее количество элементов соответствующей размерности массива. Допускается объявление массивов с размерностью не больше трёх. После того как массив будет объявлен, переопределить его невозможно (то есть оператор переопределения наподобие ReDim языка Visual Basic не поддерживается). Нумерация элементов начинается с 1;
AS	необязательное ключевое слово объявления типа данных переменной;
DataType	указатель типа данных переменной: Boolean (логический), Discrete (дискретный), Integer (целый), ElapsedTime (прошедшее время), Float (с плавающей запятой), Real (вещественный), Double (с двойной точностью), String (символьный), Message (Сообщение), Time (время), Object (Объект). Если конструкция AS будет пропущена, переменная по умолчанию объявляется переменной целого типа. Например, оператор DIM LocVar1; эквивалентен записи DIM LocVar1 AS Integer;

---

**Примечание.** В отличие от имён атрибутов, символы точки (".") в именах переменных недопустимы. Имена переменных и обозначения типов данных нечувствительны к регистру символов. Если имя объявляемой переменной совпадёт с именем другого именованного объекта в скрипте (например с именем атрибута, с именем-синонимом или именем внешнего объекта), в тексте скрипта оно будет интерпретироваться как имя переменной. В

случае совпадения с именем-синонимом при проверке синтаксиса скрипта будет выдано предупреждение о том, что имя-синоним будет проигнорировано.

Ссылка на весь массив обозначается квадратными скобками [], которые могут использоваться для обозначения как локального массива, так и атрибута-массива. Например, присвоить значения элементов атрибута-массива соответствующим элементам локального массива можно с помощью оператора следующего вида:

```
locarr[] = tag.attr[];
```

Операторы DIM могут находиться в любом месте скрипта. Вместе с тем они должны предшествовать случаям использования объявляемых переменных в любых других операторах и выражениях. Если локальная переменная будет указана в операторе, стоящем до оператора DIM, при проверке синтаксиса скрипта, выполняемой во время сохранения объекта с этим скриптом, будет выдано сообщение о необходимости определить данную переменную.

**Внимание!** Указанная проверка выполняется только при сохранении определения объекта. Она не совпадает с процедурой, запускаемой нажатием кнопки Проверка скрипта (Validate Script).

Объявление нескольких переменных одним оператором DIM не поддерживается. Каждая переменная должна быть объявлена отдельно от других. Следующие операторы являются недопустимыми:

```
DIM LocVar1 AS Integer, LocVar2 AS Real;
```

```
DIM LocVar3, LocVar4, LocVar5 AS Message;
```

При использовании объявленных переменных в правой части оператора присваивания индикатор качества объекта, стоящего слева, всегда получает значение "Good". Например, при выполнении следующего скрипта:

```
DIM x AS int;
DIM y AS int;
x = 5;
y = 5;
me.attr = 5;
me.attr = x;
me.attr = x+y;
```

качество значений me.attr в каждом случае будет "Годное" ("Good").

**Примечание.** Если переменная указана в правой части выражения, её качество всегда интерпретируется как годное, для того чтобы обеспечить определение качества значений других переменных.

**Внимание!** Указывать пользовательские атрибуты в вызовах системных объектов в виде аргументов нельзя. Чтобы передать системному объекту значение пользовательского атрибута, нужно предварительно присвоить его значение локальной переменной и затем указать её в аргументах вызова, или же в вызове объекта явным образом указать преобразование пользовательского атрибута в строку символов с помощью соответствующей функции.

## Числовые и символьные значения

Константы целого типа должны указываться в следующем формате:

```
..... = 0 ... [.....] <.....>
<.....>*
```

где

..... ::= + | -

ненулевая цифра ::= 1-9

..... ::= 0-9

То есть целая константа представляет собой число 0 или число, состоящее из одной или нескольких цифр, перед которыми может стоять знак. Нули в качестве первой цифры многозначного числа не допускаются. Указание константы вне диапазона от -2147483648 до +2147483647 приводит к ошибке переполнения.

Указание в начале числа символов "0x" или "0X" приводит к тому, что число трактуется как шестнадцатеричная константа. Дополнительно может быть также указан знак числа "+" или "-".

Формат записи шестнадцатеричных констант следующий:

..... = [<.....> <0><x [.....]> <.....>\*]

где

..... ::= + | -

..... ::= 0-9, A-F, a-f (.....  
..... 8 ..... (32-  
.....))

Константы с плавающей запятой должны указываться в следующем формате:

..... = [<.....> <.....>\* .  
<.....> + [<.....>]]

или

[<.....> <.....> + [ . <.....>\* [<.....> ] ]

где

..... ::= + | -

..... ::= 0-9 (.....  
.....)

..... ::= . (.....) , .....  
.....

Константы с плавающей запятой могут указываться как значения переменных float, real и double. Если переменной будет присваиваться слишком большое значение, модуль грамматического разбора при проверке синтаксиса скрипта выдаст сообщение о переполнении.

Если перед символом точки нет цифр, после неё должна стоять хотя бы одна цифра. Если нет ни точки, ни экспоненциальной части, положение точки подразумевается непосредственно за последней цифрой в записи числа.

Если имя атрибута совпадает с записью вещественной константы в экспоненциальной форме (например 5E3), его нужно указывать вместе с квалификатором атрибута следующим образом:

Attribute ("5E3")

Строки символов должны записываться следующим образом: все символы должны быть заключены между символами двойных кавычек. Эта запись называется "строка в кавычках". Двойные кавычки внутри строки записываются как пара двойных кавычек. Например, предложение:

Joe said, "Look at me."

на языке QuickScript .NET в виде строки символов должно быть записано как:

"Joe said, "Look at me"."

## Управляющие структуры языка QuickScript .NET

В языке QuickScript .NET поддерживаются следующие четыре управляющие структуры:

- IF ... THEN ... ELSEIF ... ELSE ... ENDIF
- цикл FOR ... TO ... STEP ... NEXP
- FOR EACH ... IN ... NEXT
- WHILE ... ENDWHILE

Описание каждой из них приведено в следующих параграфах.

### IF ... THEN ... ELSEIF ... ELSE ... ENDIF

Конструкция IF-THEN-ELSE-IF используется для выполнения указанных операторов в зависимости от значения логического условия. Формат записи:

```
IF <выражение> THEN
    [операторы]
[ { ELSEIF
    [операторы] } ]
[ ELSE
    [операторы] ]
ENDIF;
```

Значение <выражения> определяется как значение логического выражения в соответствии со следующей таблицей:

Тип результата, возвращаемого <выражением>	Интерпретация результата
Boolean, Discrete	Преобразование не требуется.
Integer	"0" интерпретируется как "False", ненулевые значения – как "True".
Float, Real	"0" интерпретируется как "False", ненулевые значения – как "True".
Double	"0" интерпретируется как "False", ненулевые значения – как "True".
String, Message	Преобразование невозможно. Использование выражения, возвращающего результат этих типов, приводит к генерации ошибки при проверке правильности скрипта.
Time	Преобразование невозможно. Использование выражения, возвращающего результат данного типа, приводит к генерации ошибки при проверке правильности скрипта.
ElapsedTime	Преобразование невозможно. Использование выражения, возвращающего результат данного типа, приводит к генерации ошибки при проверке правильности скрипта.
Object	Использование выражения, возвращающего результат Object, возможно. Но во время исполнения этот объект

	будет преобразован к логическому типу. При невозможности преобразования во время работы программы будет сгенерирована исключительная ситуация.
--	--

Если <выражение> принимает значение "True", выполняется первый блок операторов, указанный после ключевого слова THEN. В противном случае выполняется второй блок операторов после ключевого слова ELSE (если он существует). Конструкция ELSEIF обеспечивает последовательную проверку нескольких логических условий и является аналогией операторов-переключателей (switch), имеющихся в других языках программирования.

Примеры:

```
IF value = 0 THEN
    Message = "Value is zero";
ELSEIF value > 0 THEN
    Message = "Value is positive";
ELSEIF value < 0 THEN
    Message = "Value is negative";
ELSE
{.....}
ENDIF
```

Кроме того, поддерживается следующий синтаксис:

```
IF <....._.....> THEN
    [.....]
[ { ELSEIF
    [.....] } ]
[ ELSE
    [.....] ]
ENDIF;
ENDIF;
```

Эта конструкция представляет собой вложенный оператор IF, требующий дополнительного ENDIF.

## Цикл For ... TO ... STEP ... NEXT

Цикл FOR-NEXT используется для многократного выполнения функции или блока функций во время однократного исполнения скрипта, содержащего цикл. Общий формат записи цикла выглядит следующим образом:

```
FOR <....._.....> = <....._.....> TO
<....._.....>
    [STEP <.....>]
    [.....]
    [EXIT FOR;]
    [.....]
NEXT;
```

Здесь:

- *переменная\_цикла* должна быть переменной Integer, Float, Real или Double.

- *начальное\_значение* должно представлять собой допустимое выражение, определяющее начальное значение *переменной\_цикла* до выполнения цикла.
- *конечное\_значение* должно представлять собой допустимое выражение. Как только значение *переменной\_цикла* станет больше этой величины, выполнение цикла прекращается и управление передаётся оператору, следующему непосредственно за оператором NEXT.
- Указанное справедливо, если приращение задаёт увеличение значения *переменной\_цикла*. Если же оно задаёт уменьшение, выполнение цикла прекращается, как только значение *переменной\_цикла* становится меньше *конечного\_значения*.
- Приращение, это величина, на которую изменяется значение *переменной\_цикла* после выполнения оператора NEXT. Приращение может быть как положительной, так и отрицательной величиной. Если оно положительное, *начальное\_значение* должно быть меньшим, чем *конечное\_значение*, или должно быть равным ему. В противном случае операторы внутри цикла исполняться не будут. Если приращение отрицательно, *начальное\_значение* должно быть больше, чем *конечное\_значение*, или должно быть равным ему. Если конструкция STEP отсутствует в записи оператора цикла, приращение предполагается равным 1. Операторы составляют "тело цикла".

Возможность "досрочного" прекращения исполнения цикла обеспечивается конструкцией EXIT FOR.

Цикл FOR-NEXT исполняется следующим образом:

1. *Переменной\_цикла* присваивается *начальное\_значение*.
2. Система проверяет, превышает ли значение *переменной\_цикла* указанное *конечное\_значение*. Если да, цикл выполнен. (Если приращение является отрицательной величиной, система проверяет, имеет ли *переменная\_цикла* значение, меньшее *конечного\_значения*.)
3. Выполняются операторы, указанные в теле цикла. Выполнение цикла может быть завершено досрочно путём исполнения оператора EXIT FOR;
4. Значение *переменной\_цикла* изменяется на величину приращения (или на 1, если приращение не указано).
5. Повторяются операции шагов 2 – 4.

---

**Примечание.** Циклы FOR-NEXT могут быть вложенными. Допустимое количество вложенных циклов зависит от объёма оперативной памяти и мощности вычислительных ресурсов системы.

---

О способах использования циклов данного типа см. параграф "Примеры скриптов".

## Цикл FOR EACH ... IN ... NEXT

Оператор цикла FOR EACH могут использоваться только с коллекциями объектов, поддерживаемых серверами OLE Automation. Он позволяет выполнять одни и те же действия с каждым элементом коллекции. Общий формат операторов следующий:

```
FOR EACH <....._.....> IN <....._.....>
    [ ..... ]
    [ EXIT FOR; ]
    [ ..... ]
NEXT;
```

Здесь:

- *объект\_переменная* представляет собой переменную, тип которой поддерживается COM-интерфейсом;
- *объект\_коллекция* представляет собой переменную, содержащую коллекцию.

Как и в случае оператора FOR-NEXT, досрочное прекращение выполнения цикла FOR-EACH также возможно при помощи оператора EXIT FOR.

О способах использования циклов данного типа см. параграф "Примеры скриптов".

## Цикл WHILE

Оператор цикла WHILE используется для многократного выполнения функции или набора функций во время однократного исполнения содержащего этот цикл скрипта. Общий формат записи цикла выглядит следующим образом:

```
WHILE <.....>  
    [ ..... ]  
    [ EXIT WHILE; ]  
    [ ..... ]  
ENDWHILE;
```

Здесь <условие> представляет собой выражение, логическое значение которого определяется так же, как и в конструкции IF-THEN.

Досрочное прекращение исполнения цикла осуществляется при помощи оператора EXIT WHILE.

Цикл WHILE выполняется следующим образом:

1. Система определяет выполнение <условия>. Если оно не выполняется (значение условия равно "FALSE"), цикл считается завершённым.
2. Выполняются операторы в теле цикла. Досрочно завершить выполнение цикла можно с помощью оператора EXIT WHILE.
3. Шаги 1 и 2 повторяются.

---

**Примечание.** Циклы WHILE могут быть вложенными. Количество вложенных циклов определяется объёмом имеющейся оперативной памяти и мощности вычислительных ресурсов компьютера.

---

О способах использования циклов данного типа см. параграф "Примеры скриптов".

## Распознавание ключевых слов и идентификаторов

Тип идентификатора (то есть указание, является ли он ключевым словом или именем переменной или атрибута) определяется в следующем порядке:

- Ключевые слова.
- Имена атрибутов.

## Примеры скриптов

В настоящем параграфе содержатся примеры на языке QuickScript .NET, которые можно использовать для разработки собственных скриптов.

---

**Внимание!** Примеры иллюстрируют лишь правильное употребление операторов языка. Их выполнение может зависеть от наличия ресурсов и параметров прикладной системы.

---

Скрипты примеров выполняют следующие действия:

- Запись текстового файла на диск.
- Чтение текстового файла с диска.
- Вывод числовых значений в формате 'Picture' .NET.
- Вывод показаний времени в формате 'Picture' .NET.
- Создание XML-документа и сохранение его на диске.
- Загрузка XML-документа с диска и поиск в нём требуемой информации.
- Выдача запроса к базе данных SQL-сервера.
- Выполнение оператора INSERT с параметрами.
- Общий доступ к SQL-соединению (или другому объекту .NET).
- Создание поисковой таблицы и поиск информации.
- Инициализация и использование символьного массива.
- Инициализация и использование двумерного массива целого типа.
- Чтение списка каталогов в каталоге C:\.
- Отсылка электронной почты с помощью Exchange.
- Отсылка электронной почты по протоколу SMTP.
- Отсылка электронной почты с помощью веб-службы.
- Определение температуры в указанном районе с помощью веб-службы.
- Определение температуры в городе с помощью веб-службы.
- Открытие электронной таблицы Excel с помощью метода CreateObject().
- Доступ к электронной таблице Excel средствами библиотеки импортированных типов.
- Доступ к электронной таблице Excel пакета Office XP средствами библиотеки импортированных типов.
- Обращение к таблице Excel по протоколу DDE.
- Считывание индикатора производительности.

## Запись текстового файла на диск

```
DIM sw AS System.IO.StreamWriter;

sw = System.IO.File.CreateText("C:\MyFile.txt");
sw.WriteLine("one");
sw.WriteLine("two");
sw.WriteLine("three");
sw.Close();
```

## Чтение текстового файла с диска

```
DIM sr AS System.IO.StreamReader;

sr = System.IO.File.OpenText("c:\MyFile.txt");
WHILE sr.Peek() > -1
```



```
    LogMessage(sr.ReadLine());  
ENDWHILE;  
sr.Close();
```

## Вывод числовых значений в формате 'Picture' .NET

```
DIM i AS integer;  
  
i = 1234;  
LogMessage("Total cost: " +  
i.ToString("$#,###,###.00"));
```

## Вывод показаний времени в формате 'Picture' .NET

```
DIM t AS time;  
  
t = Now();  
LogMessage("The current time is: " +  
t.ToString("hh:mm:ss") + ".");
```

## Создание XML-документа и сохранение его на диске

```
DIM doc AS System.Xml.XmlDocument;  
DIM catalog AS System.Xml.XmlElement;  
DIM book AS System.Xml.XmlElement;  
DIM title AS System.Xml.XmlElement;  
DIM author AS System.Xml.XmlElement;  
DIM lastName AS System.Xml.XmlElement;  
DIM firstName AS System.Xml.XmlElement;  
  
' ..... XML-.....  
doc = new System.Xml.XmlDocument;  
catalog = doc.CreateElement("catalog");  
doc.AppendChild(catalog);  
  
' .....  
book = doc.CreateElement("book");  
title = doc.CreateElement("title");  
author = doc.CreateElement("author");  
lastName = doc.CreateElement("lastName");  
firstName = doc.CreateElement("firstName");  
author.AppendChild(lastName);  
author.AppendChild(firstName);  
book.AppendChild(title);  
book.AppendChild(author);
```

```

catalog.AppendChild(book);
book.SetAttribute("isbn", "0385503822");
title.InnerText = "The Summons";
lastName.InnerText = "Grisham";
firstName.InnerText = "John";

' .....
book = doc.CreateElement("book");
title = doc.CreateElement("title");
author = doc.CreateElement("author");
lastName = doc.CreateElement("lastName");
firstName = doc.CreateElement("firstName");
author.AppendChild(lastName);
author.AppendChild(firstName);
book.AppendChild(title);
book.AppendChild(author);
catalog.AppendChild(book);
book.SetAttribute("isbn", "044023722X");
title.InnerText = "A Painted House";
lastName.InnerText = "Grisham";
firstName.InnerText = "John";

' ..... XML-.....
doc.Save("c:\catalog.xml");

```

## Загрузка XML-документа с диска и поиск в нём требуемой информации

```

DIM doc AS System.Xml.XmlDocument;
DIM node AS System.Xml.XmlNode;

doc = new System.Xml.XmlDocument;
doc.Load("c:\catalog.xml");

' ..... , ... ISBN .....
    044023722X
node =
doc.SelectSingleNode("/catalog/book[@isbn='044023722X']
/title");
LogMessage(node.InnerText);

' ..... , .....
    Grisham
FOR EACH node IN
doc.SelectNodes("/catalog/book[author/lastName='Grisha
m']/title")
LogMessage(node.InnerText);

```

```
NEXT;
```

## Выдача запроса к базе данных SQL-сервера

```
DIM connection AS System.Data.SqlClient.SqlConnection;  
DIM Command AS System.Data.SqlClient.SqlCommand;  
DIM reader AS System.Data.SqlClient.SqlDataReader;
```

```
connection = new  
    System.Data.SqlClient.SqlConnection("server  
    =(local);uid=sa;dataBase=northwind");
```

```
connection.Open();
```

```
Command = new System.Data.SqlClient.SqlCommand("select  
    * from customers", connection);
```

```
reader = Command.ExecuteReader();
```

```
WHILE reader.Read()
```

```
    LogMessage(reader("CompanyName"));
```

```
ENDWHILE;
```

```
reader.Close();
```

```
connection.Close();
```

## Выполнение оператора INSERT с параметрами

```
DIM connection AS System.Data.SqlClient.SqlConnection;
```

```
DIM Command AS System.Data.SqlClient.SqlCommand;
```

```
DIM regionId AS System.Data.SqlClient.SqlParameter;
```

```
DIM regionDesc AS System.Data.SqlClient.SqlParameter;
```

```
DIM CommandText AS string;
```

```
connection = new  
    System.Data.SqlClient.SqlConnection("server  
    =(local);uid=sa;dataBase=northwind");
```

```
connection.Open();
```

```
CommandText = "INSERT INTO Region (RegionID,  
    RegionDescription) VALUES (@id, @desc)";
```

```
Command = new  
    System.Data.SqlClient.SqlCommand(CommandText,  
    connection);
```

```
regionId = Command.Parameters.Add("@id",  
    System.Data.SqlDbType.Int, 4);
```

```
regionDesc = Command.Parameters.Add("@desc",  
    System.Data.SqlDbType.NChar, 50);
```

```
Command.Prepare();
```

```
regionId.Value = 5;
```

```

regionDesc.Value = "Europe";
Command.ExecuteNonQuery();

regionId.Value = 6;
regionDesc.Value = "South America";
Command.ExecuteNonQuery();

connection.Close();

```

## Общий доступ к SQL-соединению (или другому объекту .NET)

**Примечание.** Начинать нужно с создания показанной ниже библиотеки C# и её импортирования в Galaxu. При этом должен быть установлен инструментальный пакет разработки .NET SDK.

```

set path=%path%;C:\Program Files\Microsoft Visual
  Studio .NET\FrameworkSDK\Bin
csc /target:library ObjectCache.cs

```

----- ObjectCache.cs-----

```

using System;
using System.Collections;

public class ObjectCache
{
    public static void Add(string ObjectName, Object o)
    {
        Objects[ObjectName] = o;
    }

    public static void Remove(string ObjectName)
    {
        Objects.Remove(ObjectName);
    }

    public static Object Get(string ObjectName)
    {
        return Objects[ObjectName];
    }

    private static Hashtable Objects =
        Hashtable.Synchronized(new Hashtable());
}

```

----- ObjectCache.cs-----

Затем выполнить в UserDefined\_001 следующее:

```
DIM connection AS System.Data.SqlClient.SqlConnection;
```

Startup:

```
connection = new  
    System.Data.SqlClient.SqlConnection("server  
    =(local);uid=sa;dataBase=northwind");  
connection.Open();  
ObjectCache.Add("NorthwindConnection", connection);
```

Shutdown:

```
ObjectCache.Remove("NorthwindConnection");  
connection.Close();
```

Затем выполнить следующее в UserDefined\_002, UserDefined\_003 и т.д.:

```
DIM connection AS System.Data.SqlClient.SqlConnection;  
connection = ObjectCache.Get("NorthwindConnection");  
IF connection <> null THEN  
System.Threading.Monitor.Enter(connection);
```

' используем соединение

```
System.Threading.Monitor.Exit(connection);  
ENDIF;
```

## Создание поисковой таблицы и поиск информации

```
DIM zipcodes AS System.Collections.Hashtable;
```

```
zipcodes = new System.Collections.Hashtable;  
zipcodes["Irvine"] = 92618;  
zipcodes["Mission Viejo"] = 92692;  
LogMessage(zipcodes["Irvine"]);
```

## Инициализация и использование символьного массива

```
DIM Numbers[3] AS string;  
DIM s AS string;
```

```
Numbers[1] = "one";  
Numbers[2] = "two";  
Numbers[3] = "three";
```

```

LogMessage (Numbers [3] );

FOR each S in Numbers []
LogMessage (s) ;
NEXT;

```

### Инициализация и использование двумерного массива целого типа

```

DIM x[2,3] AS integer;
DIM i AS integer;

```

```

x[1, 1] = 1;
x[1, 2] = 2;
x[1, 3] = 3;
x[2, 1] = 4;
x[2, 2] = 5;
x[2, 3] = 6;

```

```

LogMessage (x[2, 3] );

```

```

FOR EACH i IN x []
LogMessage (i) ;
NEXT;

```

### Чтение списка каталогов в каталоге C:\

```

DIM dir AS System.IO.DirectoryInfo;

```

```

FOR EACH dir IN
    System.IO.DirectoryInfo ("c:\").GetDirectories ()
LogMessage (dir.FullName) ;
NEXT;

```

### Отсылка электронной почты с помощью Exchange

```

DIM session AS Object;
DIM msg AS Object;

```

```

session = CreateObject ("MAPI.Session");
session.Logon ("Employee");
msg = session.Outbox.Messages.Add ();
msg.Recipients.Add ("<.....>");
msg.Subject = ".....";
msg.Text = ".....";
msg.Send ();
session.Logoff ();

```

## Отсылка электронной почты по протоколу SMTP

```
System.Web.Mail.SmtpMail.Send
(
{from: } "<..... .. email-.....
.....>",
{to: } "<..... .. email-.....
.....>",
{subject: } "..... ..",
{body: } "."
);
```

## Отсылка электронной почты с помощью Web-службы

```
' ..... -.....
(..... .Net SDK .....).
' ..... , ..... • DOS
.....:
' set path=%path%;C:\Program Files\Microsoft Visual
Studio .NET\FrameworkSDK\Bin
' wsdl /namespace:SendMail
http://www.xmlwebservices.net/services/messaging/s
mtp\_mail/mailsender.asmx
' csc /target:library Message.cs
' .....
Message.dll • Galaxy.
' .....:
```

```
DIM m AS SendMail.Message;
```

```
m = new SendMail.Message;
m.SendSimpleMail
(
{to: } "< ..... email-.....
..... >",
{from: } "< ..... email-.....
.....>",
{subject: } ".....",
{body: } "• ..... ."
);
```

## Определение температуры в указанном районе с помощью веб-службы

```
' ..... -.....
..... me.zipcode ..... , ..... -
..... me.temperature
.....
' ..... -
..... (..... .Net SDK .....).
```

```
' ..... , ..... • DOS
  .....:
' set path=%path%;C:\Program Files\Microsoft Visual
  Studio.NET\FrameworkSDK\Bin
' wsdl
http://www.vbws.com/services/weatherretriever.asmx
' csc /target:library WeatherRetriever.cs
' .....
  WeatherRetriever.dll • Galaxy.
' .....:
```

```
DIM wr AS WeatherRetriever;
```

```
wr = new WeatherRetriever;
me.temperature = wr.GetTemperature(me.zipcode);
```

### Определение температуры в городе с помощью веб-службы

```
' ..... -.....,
' ..... ..
' ..... - .....
  me.CityState ..... (..... - .....
  ....., ..... "Los Angeles,CA"),
' ..... - ..... me.temperature
  .....
```

```
DIM request AS System.Net.WebRequest;
DIM reader AS System.IO.StreamReader;
DIM regex AS System.Text.RegularExpressions.Regex;
DIM match AS System.Text.RegularExpressions.Match;
request = System.Net.WebRequest.Create
  ("http://www.srh.noaa.gov/data/forecasts/zipcity.p
  hp?inputstring=" +
  System.Web.HttpUtility.UrlEncode(me.CityState) );
reader = new
  System.IO.StreamReader(request.GetResponse().GetRe
  sponseStream());
regex = new
  System.Text.RegularExpressions.Regex("<br>
  <br>(.*&deg;F<br>");
match = regex.Match(reader.ReadToEnd());
me.temperature = match.Groups(1);
```

### Открытие электронной таблицы Excel с помощью метода CreateObject()

```
DIM app AS Object;
DIM wb AS Object;
DIM ws AS Object;

app = CreateObject("Excel.Application");
wb = app.Workbooks.Add();
```



```
ws = wb.ActiveSheet;  
  
ws.Range("A1") = 20;  
ws.Range("A2") = 30;  
ws.Range("A3") = "=A1*A2";  
LogMessage(ws.Range("A3").Value);  
  
wb.Close(false);
```

### **Доступ к электронной таблице Excel средствами библиотеки импортированных типов**

```
DIM app AS Excel._Application;  
DIM wb AS Excel._Workbook;  
DIM ws AS Excel._WorkSheet;  
  
app = new Excel.Application;  
wb = app.Workbooks.Add();  
ws = wb.ActiveSheet;  
  
ws.get_Range("A1").Value = 1000;  
ws.get_Range("A2").Value = 1000;  
ws.get_Range("A3").Value = "=A1+A2";  
LogMessage(ws.get_Range("A3").Value);  
  
wb.Close(false);
```

### **Доступ к электронной таблице Excel пакета Office XP средствами библиотеки импортированных типов**

```
DIM app AS Excel.Application;  
DIM ws AS Excel.Worksheet;  
DIM wb AS Excel.Workbook;  
DIM a1 AS Excel.Range;  
DIM a2 AS Excel.Range;  
DIM a3 AS Excel.Range;  
  
app = new Excel._ExcelApplicationClass;  
wb = app.ActiveWorkbook;  
ws = app.ActiveSheet;  
  
a1 = ws.Range("A1");  
a2 = ws.Range("A2");  
a3 = ws.Range("A3");  
a1.Value = 1000;
```

```

a2.Value = 2000;
a3.Value = "=A1*A2";
LogMessage(a3.Value);

wb.Close(true, "c:\temp.xls" false);

```

## Обращение к таблице Excel по протоколу DDE

```

WWPoke("Excel", "sheet1", "r1c1", "Hello");
WRequest("Excel", "sheet1", "r1c1", me.Greeting);
' ..... . ..... . ..... . ..... . ..... . .....
  ..... . ..... . ..... . ..... . ..... .
WVExecute("Excel", "sheet1",
  "[SELECT("R1C1")][FONT.PROPERTIES(,"Bold")]");

```

## Считывание индикатора производительности

```

' ..... . ..... - ..... . .....
me.PercentProcessorTime ..... . .....
' ..... . .....

DIM counter AS System.Diagnostics.PerformanceCounter;

' .....

counter = new System.Diagnostics.PerformanceCounter;
counter.CategoryName = "Processor";
counter.CounterName = "% Processor Time";
counter.InstanceName = "0";

' ..... . ..... . ..... . ..... . ..... . .....
me.PercentProcessorTime = counter.NextValue();

```

## ГЛАВА 7

# Объекты-контейнеры

Контейнеры представляют собой объекты, в состав которых входят другие объекты (например реактор, в состав которого могут входить насосы, вентили и другие компоненты).

В настоящей главе рассматриваются способы применения объектов-контейнеров в приложениях IAS.

### Содержание

- Вложенные объекты
- Контейнеры ApplicationObject

## Вложенные объекты

Контейнеры можно рассматривать как один из способов иерархической организации объектов. В этом случае любое устройство может быть представлено в виде сложного объекта, состоящего из более простых устройств меньшего размера. Объекты более высокого уровня могут содержать в себе объекты низших уровней, что даёт возможность более точно моделировать производственную обстановку.

Полное имя вложенного объекта включает в себя имя объекта, содержащего вложенный. Его также можно определить как иерархическое имя объекта. Предположим, что экземпляр объекта с именем Valve1, созданного по шаблону \$Valve, содержится внутри экземпляра объекта Reactor1, созданного по шаблону \$Reactor. Предположим, что вложенное имя Valve1 было изменено на InletValve. Теперь на объект Valve1 можно ссылаться по его иерархическому имени Reactor1.InletValve. При этом в рамках иерархической структуры связей имя вложенного объекта тоже может быть изменено. Например, вложенное имя вентилля Reactor1.InletValve может быть изменено на Outlet. Имя объекта InletValve (имя тэга) не изменилось. Только его иерархическое имя упростилось в контексте связей вложения. Вложенное имя сложного объекта должно быть уникальным только в контексте его контейнера.

Объекты ApplicationObject могут содержаться внутри других объектов ApplicationObject. Объекты Atea могут входить в состав объектов только этого же типа.

## Примеры объектов-контейнеров

Как правило, контейнеры определяются пользователями по шаблону \$UserDefined и включают в себя более простые объекты, представляющие другие устройства. Например, объект Tank может содержать два объекта, представляющие впускной и выпускной клапаны (о способах использования циклов данного типа см. параграф DiscreteDevice).

В данном случае пользователь должен выполнить следующие действия:

1. Создать следующие экземпляры объектов: Tank1 по шаблону \$UserDefined и Valve1 по шаблону \$DiscreteDevice. В данный момент для клапана Valve1 определено только одно имя "Valve1".
2. В общей структуре или структуре использования перетащить с помощью мыши объект Valve в объект Tank1.

---

**Примечание.** Если в объекте Tank1 уже есть объект с вложенным именем Valve1, Galaxy сгенерирует другое вложенное именем для перемещаемого объекта, например Valve1\_1.

---

3. Изменить вложенное именем Valve1 в объекте Tank1 на Outlet. В этот момент для этого объекта уже имеется два имени: Valve1 и Tank1.Outlet.
4. Создать экземпляр объекта Reactor1 по шаблону \$UserDefined.
5. В общей структуре или структуре использования перетащить объект Tank1 в объект Reactor1.
6. Изменить вложенное имя Tank1 на Tank. После этого объект Tank1 имеет два имени: Tank1 и Reactor1.Tank. Объект Valve1 имеет трёхсоставное вложенное имя Reactor1.Tank.Outlet.

Созданные три объекта (объект Reactor1 с вложенным объектом Tank1, в котором также имеется вложенный объект Valve1) образуют следующую иерархическую структуру:

Собственное название (имя тэга)	Иерархическое имя
Reactor1	-
Tank1	Reactor1.Tank
Valve1	Reactor1.Tank.Outlet или Tank1.Outlet

Ниже дан пример создания и использования шаблона, в состав которого входит другой шаблон.

---

**Примечание.** В шаблонах имена тэгов не используются (зарезервированы для обозначения экземпляров объектов).

---

1. Пользователь создаёт производные шаблоны Tank1 на базе шаблона \$UserDefined и шаблон \$Valve – на базе шаблона \$DiscreteDevice.
2. Пользователь создаёт производный шаблон \$Inlet на базе шаблона \$Valve.

---

**Примечание.** В следующих примерах предполагается, что стандартные имена объектов, присвоенные им в момент создания, были изменены.

---

3. В панели шаблонов пользователь перетаскивает шаблон \$Inlet в шаблон \$Tank. Если в нём уже имеется шаблон с именем Inlet, система сгенерирует новое имя (например Inlet\_1).
4. Вложенный шаблон получил иерархическое имя \$Tank.Inlet.
5. Пользователь создаёт экземпляр объекта Tank1 по шаблону \$Tank.
6. В общей структуре и структуре использования появится экземпляр объекта с именем Tank1, содержащий экземпляр объекта с именем Inlet.

## Изменение вложенного имени объекта

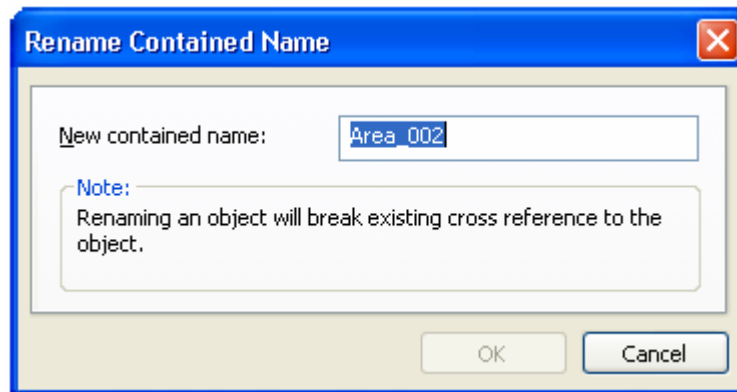
Для изменения вложенного имени объекта нужно, чтобы он в текущий момент не был захвачен и не использовался в рабочем приложении.

Новое вложенное имя должно удовлетворять требованиям по именованию объектов и не должно совпадать с вложенным именем существующего

вложенного объекта на том же уровне иерархии связей. Подробнее см. параграф "Допустимые имена и символы".

#### Чтобы изменить вложенное имя объекта

1. Выделите нужный объект в панели структуры приложения.
2. Выполните команду **Изменить вложенное название (Rename Contained Name)**. Появится окно **Изменение вложенного названия (Rename Contained Name)**.



3. Введите новое вложенное имя.
4. Щёлкните **ОК** для изменения вложенного имени или кнопку **Отмена (Cancel)** для отмены переименования. После изменения вложенного имени оно будет отражено во всех ИСР, подключенных к Galaxy.

---

**Внимание!** Ссылки на переименованный объект, используемые в других объектах, становятся недействительными. Во время работы приложения эти объекты будут получать данные с плохим (Bad) качеством.

---

## Контейнеры ApplicationObject

Объекты ApplicationObject могут входить в любой другой объект ApplicationObject.

Рассмотрим в качестве примера связей вложения резервуар, в котором имеются следующие устройства:

- Впускной клапан.
- Смеситель.
- Выпускной клапан.
- Уровнемер.

Ссылки на эти объекты могут быть выполнены в скриптах несколькими способами. Каждый объект должен иметь уникальное имя (эквивалентное уникальному обозначению на схеме предприятия), например:

- Впускной клапан: имя V101.
- Смеситель: имя A102.
- Выпускной клапан: имя V103.
- Уровнемер: имя LT104.

Скрипт общего вида для управления этим резервуаром, в шаблоне которого использовались бы подобные обозначения, невозможен. Вместе с тем можно изменить их вложенные имена на Inlet, Agitator, Outlet и Level соответственно. В контексте каждой иерархической структуры эти вложенные имена будут уникальными. Поэтому, если объект "резервуар"

называется T001, эти вложенные объекты будут именоваться следующим образом:

- T001.Inlet.
- T001.Agitator.
- T001.Outlet.
- T001.Level.

Подобное именование создаёт контекст вложения для экземпляров объектов в T001. Кроме того, пользователь получает возможность указывать в скриптах вложенные ссылки, подобные:

- Me.Outlet – для ссылки на дочерние объекты в скриптах родительского объекта.
- MyContainer.Inlet – для ссылок в скриптах дочерних объектов на другие дочерние объекты или на свойства родительского объекта.

Подобная связь вложения шаблонов существует в Панели шаблонов. Связь вложения экземпляров объектов отображается в общей структуре и структуре использования. Панель шаблонов разделена на секции. В каждой из секций шаблоны представляются в виде древовидной структуры. Если шаблон содержит другие шаблоны, рядом с его обозначением выводится специальный значок развёртывания. Количество уровней вложенности шаблонов не должно превышать 10.

---

**Примечание.** Базовые шаблоны не могут входить в состав других шаблонов, будь то контейнеры или вложенные шаблоны. Связи вложения существуют только между производными шаблонами.

---

В структуре вывода иерархия вложения не отображается. Сведения о вложенности показаны в ней следующим образом:

- Если экземпляр объекта не является вложенным, в структуре отображается имя соответствующего тэга.
- Если экземпляр объекта является вложенным, в структуре отображаются имя тэга и иерархическое имя.
- Если шаблон не является вложенным, в структуре отображается имя объекта.
- Если шаблон является вложенным, в структуре отображается его иерархическое имя.

Вложенность экземпляров объектов ограничена объектами Area, которые могут содержать другие объекты Area, а также объектами AppObject, которые могут содержать другие объекты AppObject.

Допускается изменение как собственного имени экземпляра объекта (имени тэга), так и вложенного имени. Шаблон может иметь только одно имя шаблона.

## ГЛАВА 8

# ССЫЛКИ

Ссылки являются основным инструментом идентификации, а также обмена данными между объектами в ArchestrA. Каждый объект, атрибут или свойство обозначается уникальным способом, и именно эти обозначения передаются системой обмена сообщениями ArchestrA.

В настоящей главе даётся краткое описание ссылок и методов их использования при разработке приложений IAS.

## Содержание

- Обмен сообщениями
- Виды ссылок
- Ссылки и перекрёстные ссылки
- Окно свойств объекта
- Окно поиска объектов
- Просмотр тэгов и ссылок Galaxy из InTouch

## Обмен сообщениями

Каждый объект обладает определённым набором свойств, таких как Value (Значение) или Quality (Качество). Все операции чтения или записи данных, выполняемые системой обмена сообщениями ArchestrA, являются по своей природе предопределёнными. Иными словами, если какое-либо действие не может быть выполнено, затребовавший его выполнение клиент будет извещён о возникновении проблемы.

Обмен сообщениями характеризуется следующими характеристиками:

- Гарантированный отклик.
- Сигнатуры названий.
- Статус.
- Качество данных.
- Сохранение порядка сообщений в структуре приоритетов.
- Буферизация данных в операциях обмена сообщениями между объектами AppEngine (создание копий).
- Синхроимпульсы "публикация-подписка".

## Виды ссылок

Ссылки определяют объект или значение одного из его атрибутов. Строка ссылки состоит из обозначения объекта и обозначения его атрибута:

Ссылка = ссылка на объект AutomationObject + ссылка на атрибут.

Например, в ссылке TIC101.PV строка "TIC101" представляет собой обозначение объекта AutomationObject, строка "PV" – обозначение его атрибута.

Фактически ссылка представляет собой имя объекта и имя атрибута:

Объект.Атрибут

Если в ссылке имя атрибута отсутствует, подразумевается атрибут PV.

---

**Примечание.** Атрибут PV имеется не у всех объектов.

---

Ссылки представляют собой сочетание строк длиной не более 32 символов, разделённых символами точки. Точки внутри подстроки ссылки недопустимы. Использовать знаки математических операций в ссылках нельзя. Кроме того, в каждой подстроке ссылки должен присутствовать хотя бы один нецифровой символ.

Относительные ссылки вроде "Me" допустимы. Допустимая ссылка должна содержать или относительную ссылку, или хотя бы одну подстроку.

Следующие относительные ссылки являются разрешёнными: "Me", "MyContainer", "MyArea", "MyPlatform", "MyEngine". Применение относительных ссылок имеет большое значение при разработке скриптов, поскольку абсолютные ссылки не всегда допустимы.

Следующие ссылки являются обозначениями свойств: ".Name", ".Value", ".Type", ".Quality". Если название свойства совпадает с зарезервированным словом Archestra, его применение возможно, но в этом случае в строке должно быть использовано ключевое слово PROPERTY в формате PROPERTY(название\_свойства). В остальных случаях указание ключевого слова PROPERTY (допускается запись в любом регистре) необязательно.

Если в ссылке не указано никакое свойство, подразумевается свойство Value.

В ссылке может быть указано свойство Value атрибута типа массив, дополненное указанием размерности (не более одной):

- [i] – для определения отдельного элемента.
- [] – для определения массива в целом.

Символ "i" обозначает константу целого типа.

При использовании относительных ссылок, таких как "MyContainer", можно ссылаться на объект, вложенный в этот контейнер. Например, ссылка "MyContainer.InletValve.PV" будет эквивалентна ссылке "Tank1.InletValve.PV" в следующей структуре:

```
Tank1
  InletValve
  OutletValve
```

## Формат ссылок

В определении структурных элементов ссылок используются следующие символы:

::= (означающий "может быть заменено на").

| (означающий "или").

[] (означающий необязательный элемент).

{ } (означающий, что заключённый в скобки элемент может быть опущен или использован однократно или многократно).

Элементы внутри угловых скобок "<>" представляют собой литералы.



- ссылка ::= <ссылка\_на\_объект><ссылка\_на\_атрибут> | <имя\_тэга>.
- ссылка\_на\_объект ::= <абсолютная\_ссылка> | <относительная\_ссылка>
- абсолютная\_ссылка ::= <имя\_тэга>{.<вложенное\_имя>}
- имя\_тэга ::= <идентификатор>
- вложенное\_имя ::= <идентификатор>
- относительная\_ссылка ::= <относительное\_имя> | <относительная\_вложенная\_ссылка>
- относительная\_вложенная\_ссылка ::= MyContainer.<вложенное\_имя> | MyArea.<вложенное\_имя>
- относительное\_имя ::= Me | MyContainer | MyArea | MyHost | MyEngine | MyPlatform
- ссылка\_на\_атрибут ::= <ссылка\_на\_значение> | <ссылка\_на\_свойство>
- обозначение\_атрибута ::= [.<примитив>][.<атрибут>]
- ссылка\_на\_значение ::= <обозначение\_атрибута>[<индекс\_массива>]
- индекс\_массива ::= <открывающая\_скобка>{<порядковый\_номер>}<закрывающая\_скобка>[<открывающая\_скобка><порядковый\_номер><закрывающая\_скобка>][<открывающая\_скобка><порядковый\_номер><закрывающая\_скобка>]
- ссылка\_на\_свойство ::= <обозначение\_атрибута>.<свойство>
- свойство ::= Value|Type|Quality|разряд|Dimension|SecurityClassification|Locked|Category|определение\_свойства
- определение\_свойства ::= PROPERTY(Value|Type|Quality|BitField|Dimension|SecurityClassification|Locked|Category)
- разряд ::= .00, .01, .02, ..., .31 (действителен только для атрибутов MxInteger; в противном случае во время работы приложения возникнет ошибка конфигурирования)
- атрибут ::= <статический\_атрибут> | <динамический\_атрибут>
- статический\_атрибут ::= [<статический\_атрибут>.<идентификатор>
- динамический\_атрибут ::= <любой\_символ\_за\_исключением\_символов\_пропуска>{<любой\_символ\_за\_исключением\_символов\_пропуска>}
- примитив ::= [<примитив>.<идентификатор>
- идентификатор ::= <допустимый\_символ>{<допустимый\_символ>}
- допустимый\_символ ::= <буква> | <цифра> | <специальный\_символ>
- буква ::= любая буква любого национального алфавита
- цифра ::= любой числовой символ
- специальный\_символ ::= любой графический знак за исключением следующих:  
 . + - \* /\ = ( ) ` ~ ! % ^ & @ [ ] { } | : ; ' , < > ? " символы\_пропуска
- символы\_пропуска ::= CR, LF, Tab, пробел, FF (возвращаемые функцией iswspace())

- любой\_символ\_за\_исключением\_символов\_пропуска ::= любой символ за исключением символов\_пропуска
- открывающая\_скобка ::= [
- закрывающая\_скобка ::= ]
- указание символов кавычек в именах тэгов, примитивов и атрибутов недопустимо
- идентификатор\_Galaxy ::= <буква> | <цифра>

**Примечания:**

- имя\_тэга представляет собой уникальное обозначение объекта.
- вложенное\_имя представляет собой (необязательное) вложенное имя объекта, которое может быть указано в ссылке, когда нужный объект входит в состав другого объекта.
- порядковый\_номер – "-1" или любое целое положительное число от 1 до 32767.
- Длина идентификатора не должна превышать 32 символов.
- Имя атрибута и название примитива могут содержать несколько идентификаторов. Длина каждого идентификатора не должна превышать 32 символов. Каждый идентификатор отделяется точкой (.). Общая длина названия примитива или атрибута (как статического, так и динамического) не должна превышать 329 символов.
- относительное\_имя и свойство не чувствительны к регистру используемых символов (это же справедливо и для записи ключевого слова PROPERTY).
- Если ссылка на атрибут отсутствует, предполагается атрибут ".PV". Если атрибут PV является массивом, такая ссылка будет неверной. Для атрибута типа массив требуется явное указание ".PV[]". Исключение из этого правила составляет случай, когда ссылке предшествует символ "@". Подобная ссылка представляет собой ссылку на объект, а не на его атрибут или свойство. В настоящее время такой формат ссылки используется только в панели Порядок исполнения (Execution Order) на странице Сведения об объекте (Object Information) редактора объектов.
- Не следует указывать названия свойств и названия псевдо-свойств InTouch в качестве имён атрибутов и примитивов при расширении функциональных возможностей объекта на страницах **Скрипты (Scripts), Пользовательские атрибуты (UDAs) и Расширения (Extensions)**. В число названий свойств в IAS входят следующие: Locked, Category, HasRuntimeSetHandler, Name, Type, Quality, Dimension1, Value, SecurityClassification, 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 и 31. В число названий псевдо-свойств InTouch входят следующие: #VString, #VString1, #VString2, #VString3, #VString4, #EnumOrdinal, #ReadSts, #WriteSts и #QString.

---

**Внимание!** Разрешение ссылок выполняется Galaxy. Если Репозиторий Galaxy недоступен, разрешение ссылок выполняется на уровне одноранговых узлов. После начального разрешения ссылки объекту назначается имя-синоним, определяющее местоположение объекта в сети. Если объект будет перемещён или переименован, система повторно выполняет процедуру разрешения ссылки и назначения нового имени-синонима.

---

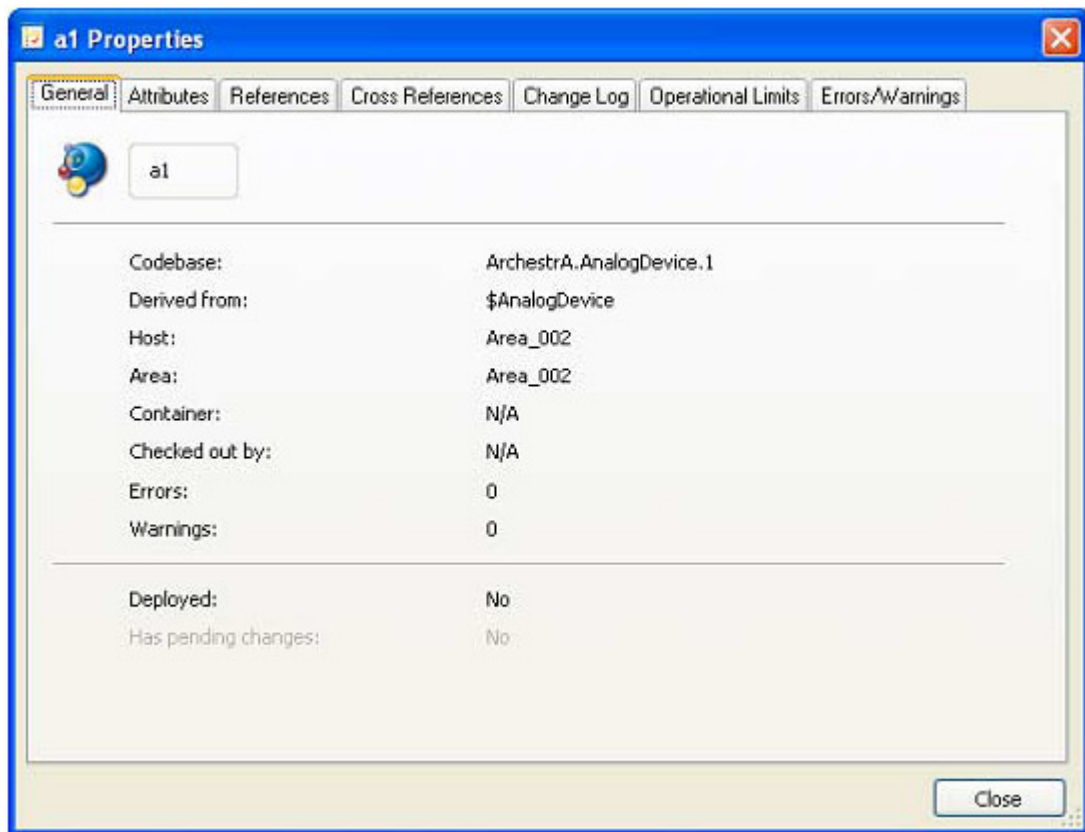
## Ссылки и перекрёстные ссылки

Определение ссылок объекта (то есть, к каким объектам он обращается) и перекрёстных ссылок (то есть какие объекты обращаются к нему) выполняется в окне свойств объекта **Свойства (Properties)**. Это окно открывается после командой **Свойства (Properties)** меню **Galaxy**.

**Примечание.** Показываемые в окне свойств объекта ссылки и перекрёстные ссылки характеризуют связи только между объектами. Связи между зонами, контейнерами или хост-системами в их число не входят.

## Окно свойств объекта

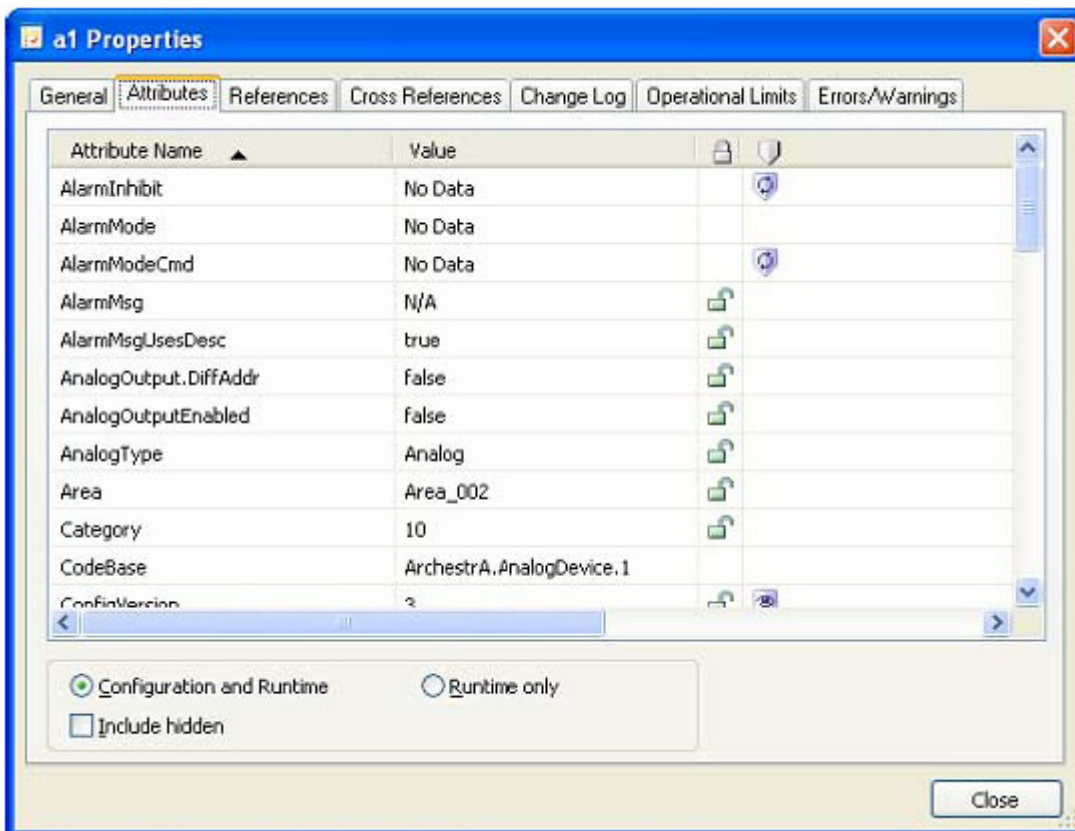
В этом окне в дополнение ко всем свойствам объекта отображаются такие его параметры, как ссылки, перекрёстные ссылки, журнал изменений, сообщения об ошибках и предупреждения. Выводимая информация может оказаться полезной для принятия решений в отношении объектов. Например, прежде чем удалить объект, нужно определить, к каким объектам он обращается и какие объекты обращаются к нему. Удаление объекта (а также переименование экземпляров, изменение вложенных имён) может привести к неверной работе приложения.



На странице с закладкой **Общие (General)** отображаются следующие сведения:

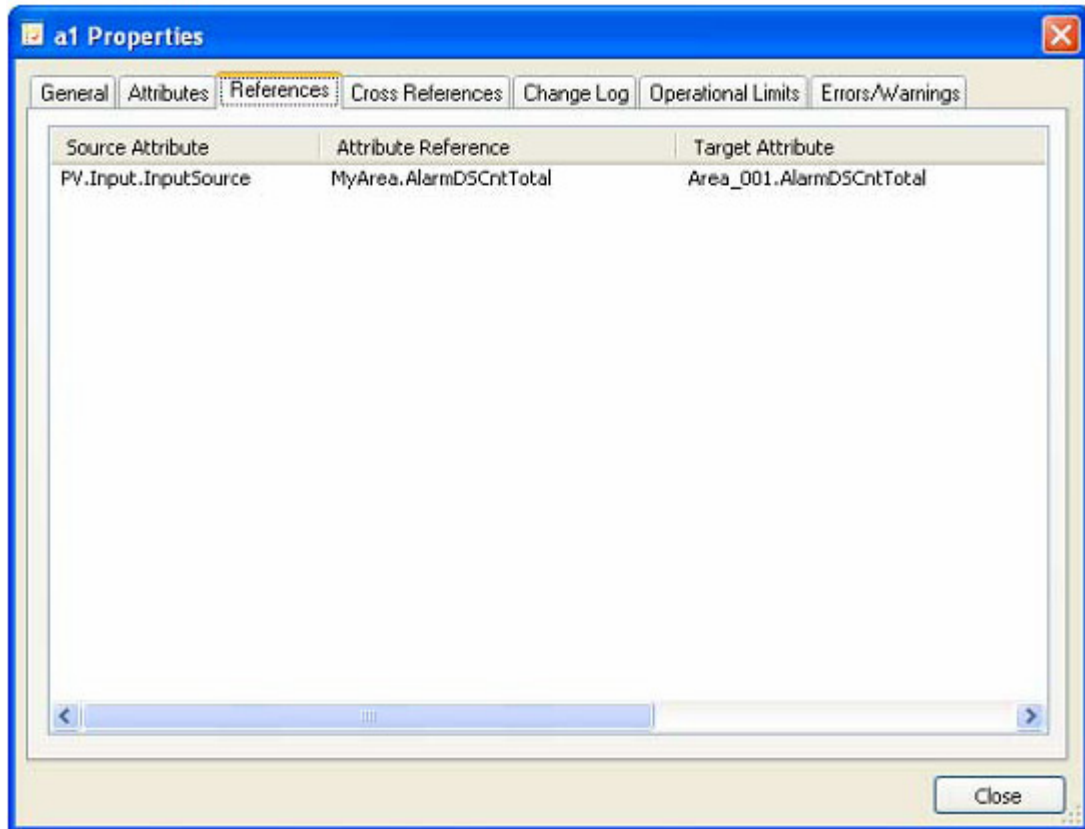
- **Обозначение кода (Codebase):** имя и версия объекта.
- **Получен из (Derived From):** имя родительского шаблона.
- **Хост-объект (Host):** объект, содержащий данный объект.
- **Зона (Area):** зона данного объекта.
- **Контейнер (Container):** объект-контейнер данного объекта.

- **Захвачен (Checked Out By)**: имя пользователя, захватившего данный объект.
- **Ошибок (Errors)**: количество ошибок выполнения действий с объектом.
- **Предупреждений (Warnings)**: количество предупреждений, выданных при выполнении действий с данным объектом.
- **Используется (Deployed)**: индикатор использования данного объекта в рабочем приложении.
- **Выполнены изменения (Has Pending Changes)**: индикатор выполнения действий над конфигурацией объекта.

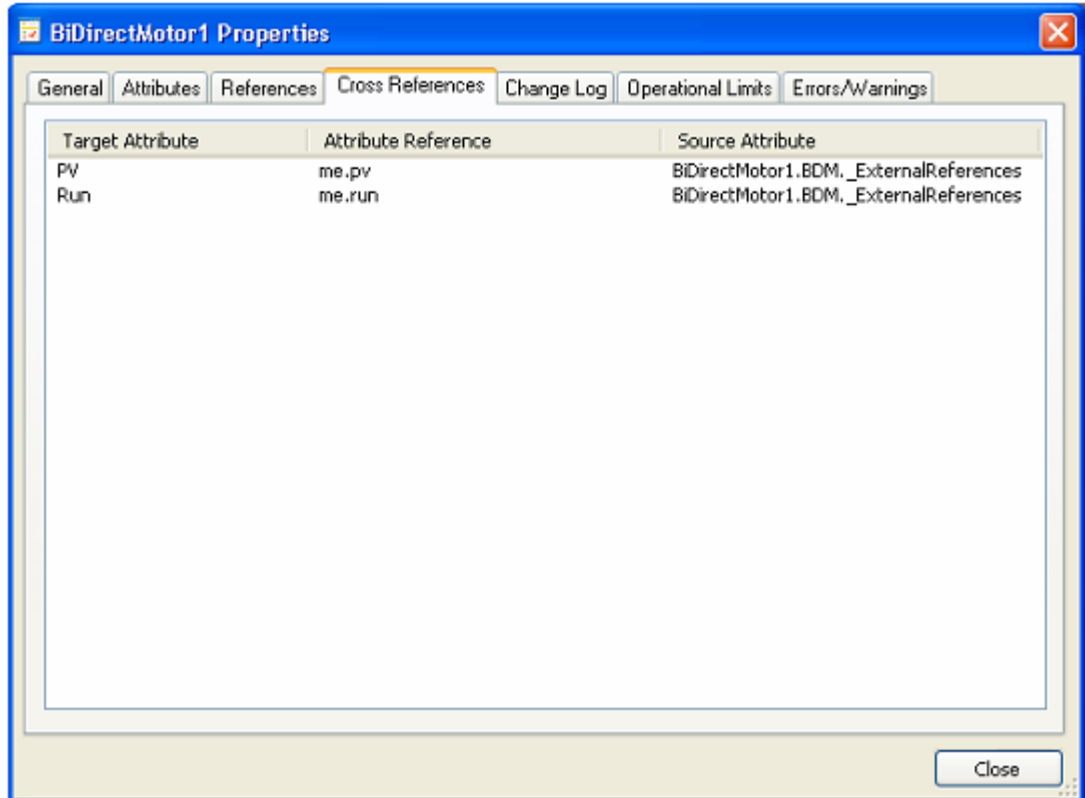


На странице с закладкой **Атрибуты (Attributes)** отображаются следующие сведения:

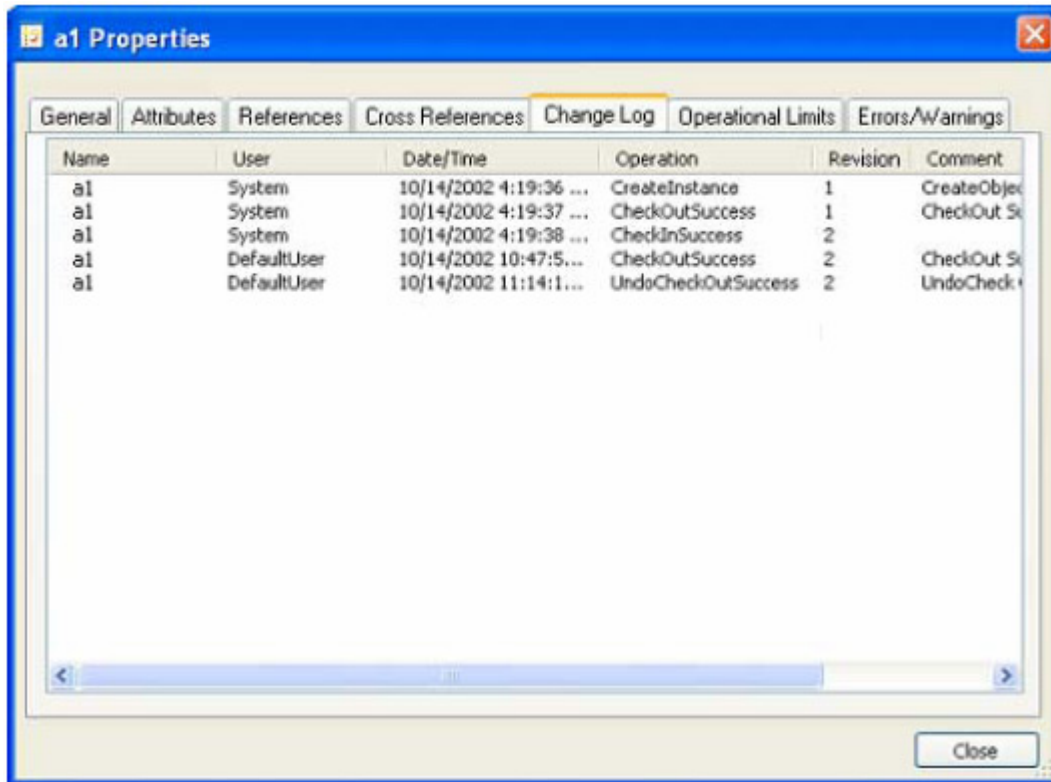
- **Имя атрибута (Attribute Name)**: перечень атрибутов объекта, а также их текущие значения и состояния блокировки и доступа. Содержание этого списка зависит от выбора переключателя **Конфигурирование и исполнение (Configuration and Runtime)** или **Только исполнение (Runtime Only)**.
- **Конфигурирование и исполнение (Configuration and Runtime)**: установите этот флажок, если в перечне должны быть отображены атрибуты объекта для режимов конфигурирования и исполнения.
- **Только исполнение (Runtime Only)**: установите этот флажок, если в перечне должны быть отображены атрибуты объекта только для режима исполнения приложения.
- **Включая скрытые (Include Hidden)**: установите этот флажок, если в перечне должны быть также отображены скрытые атрибуты объекта.



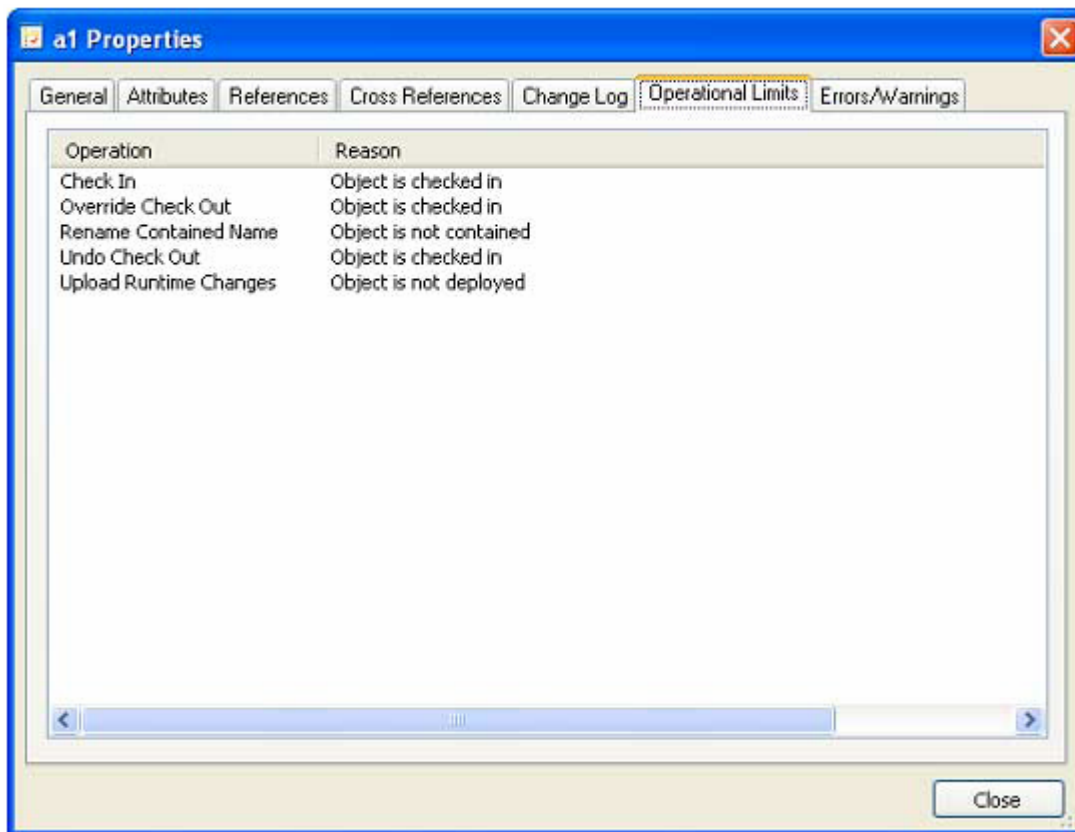
На странице с закладкой **Ссылки (References)** отображаются ссылки данного объекта на все остальные объекты Galaxy.



На странице с закладкой **Перекрёстные ссылки (Cross References)** отображаются ссылки на данный объект всех других объектов Galaxy.

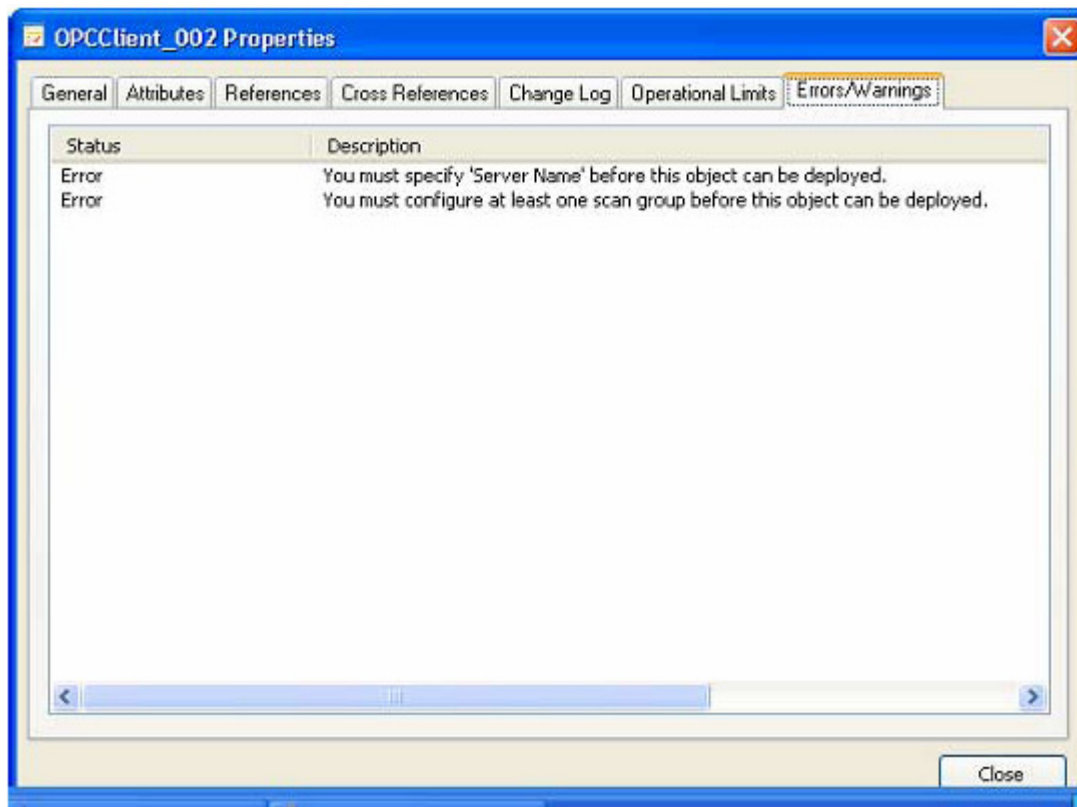


На странице **Журнал изменений (Change Log)** приведён список операций, в который использовался данный объект. В состав отображаемых сведений входит имя пользователя, выполнившего указанное действие, а также его примечания.



На странице **Оперативное состояние (Operational Limits)** отображаются сведения об оперативном состоянии объекта. Например, рассылка

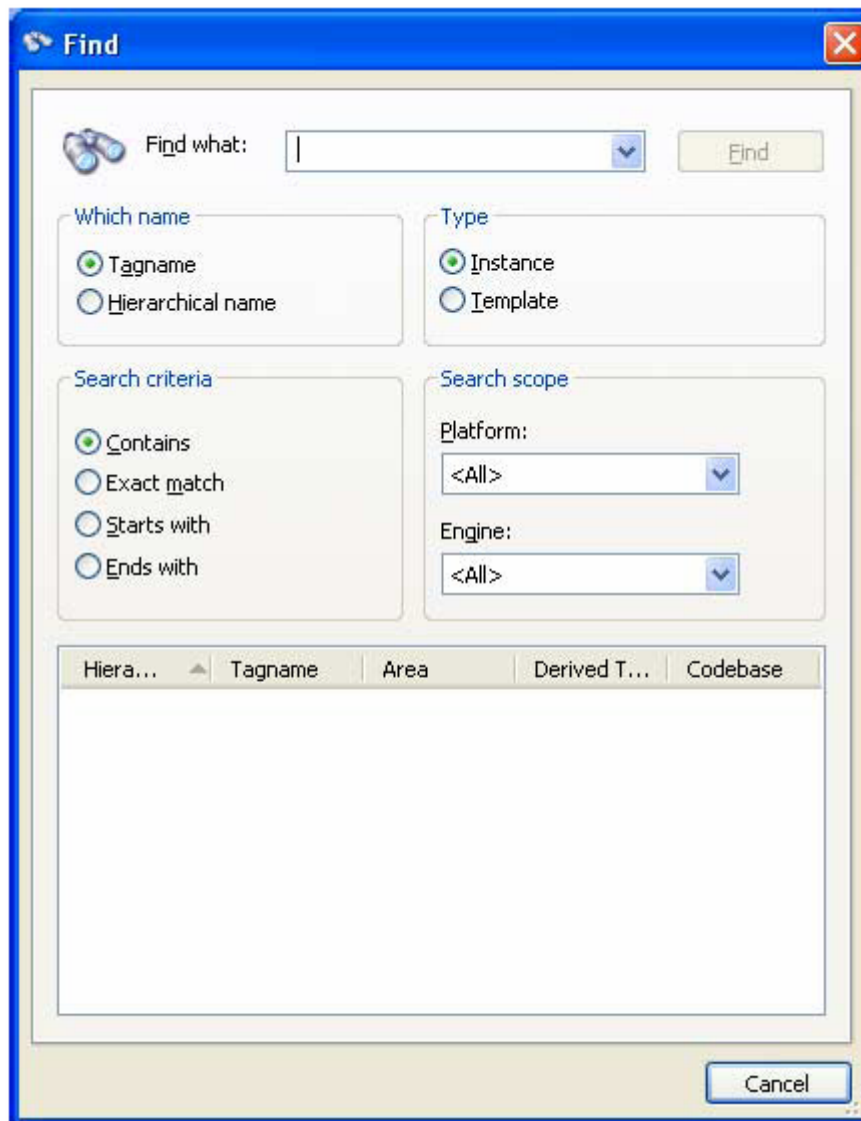
изменений конфигурации используемого объекта в данном случае не имеет смысла, поскольку объект не используется (Object is not deployed).



На странице **Ошибки и предупреждения (Errors/Warnings)** отображаются сведения о проблемах, возникавших во время конфигурирования и использования объекта.

## Окно поиска объектов

Данное окно используется для поиска по определённым критериям объектов в Galaxy.



В данном окне есть поля, с помощью которых задаются критерии поиска объектов. В поле **Поиск (Find What)** вводится имя объекта. С помощью флажков **Имя тэга (Tagname)** и **Иерархическое имя (Hierarchical Name)** задаётся тип введённого имени. В панели **Тип (Type)** указывается тип искомого объекта: **Экземпляр (Instance)** или **Шаблон (Template)**. При установленном флажке **Шаблон (Template)** будут недоступны поля панелей **Тип названия (Which Name)** и **Искать в (Search Scope)**. С помощью флажков панели **Критерий поиска (Search Criteria)** указывается, как использовать строку символов, введённую в поле **Поиск (Find What)**: **Содержит (Contains)**, **Точное совпадение (Exact match)**, **Начинается с (Starts with)**, **Заканчивается на (Ends with)**. Круг поиска определяется значениями полей панели **Искать в (Search scope)**. После того как все параметры поиска будут определены, щёлкните **Искать (Find)**. В нижней панели будут показаны результаты поиска.

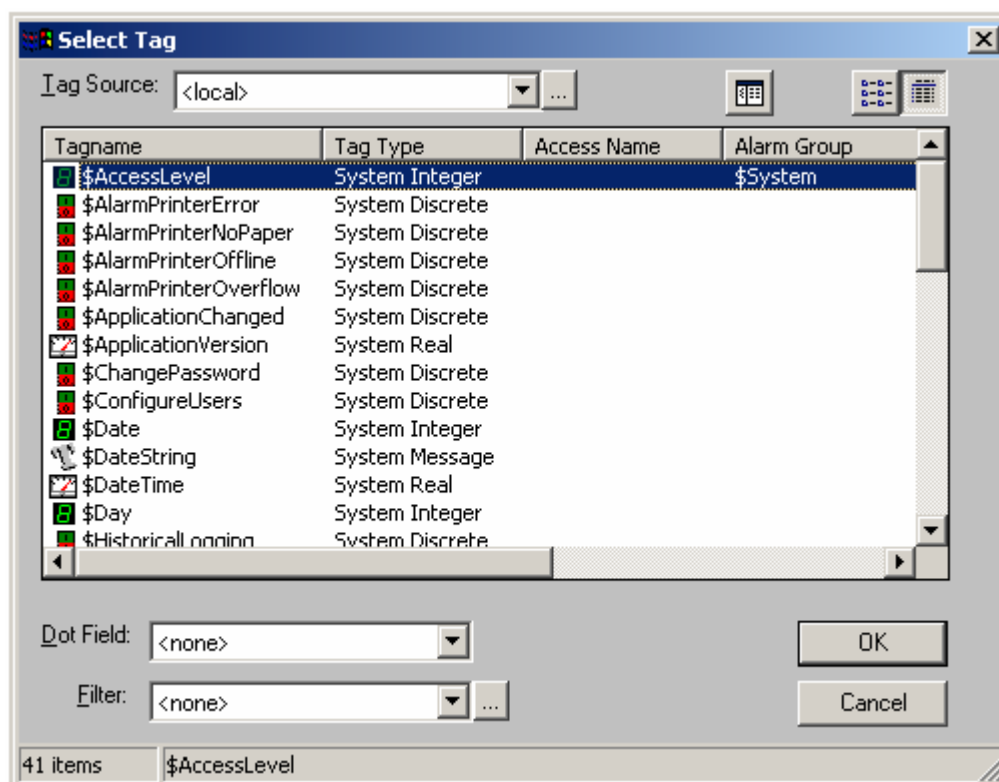
Двойной щелчок кнопкой мыши имени объекта в панели результатов приводит к его выделению в панели структуры приложения в главном окне ИСР. Если объект является резервным объектом AppEngine (подробнее см. главу "Резервирование компонентов систем Archestra"), в окне ИСР будет выведена структура использования.



## Просмотр тэгов и ссылок Galaxy из приложений InTouch

Поиск объектов Galaxy для использования в разрабатываемом приложении InTouch может быть выполнен с помощью браузера тэгов InTouch в режиме поиска без ограничений. При этом в узле InTouch должно быть установлено приложение Bootstrap и программное обеспечение ИСР. Ниже приведены способы запуска браузера тэгов InTouch в режиме поиска без ограничений:

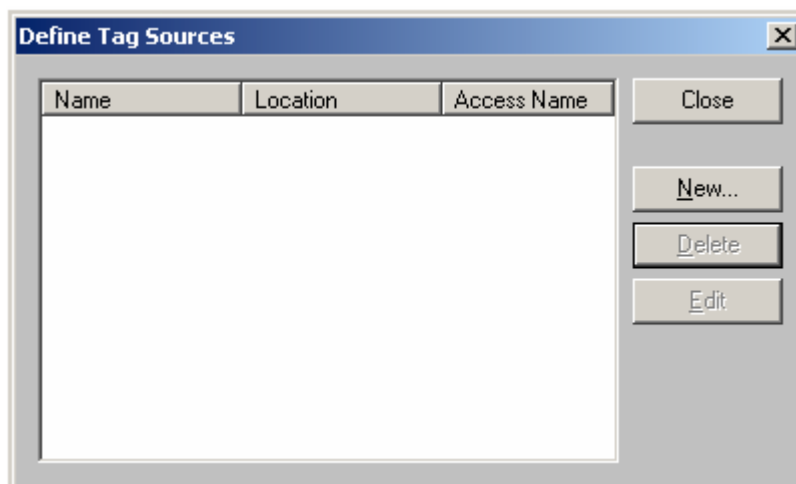
- Двойной щелчок кнопкой мыши поля ввода имени тэга или выражения анимационной функции.
- Дважды щёлкните окно ввода имени объекта ActiveX или мастер-объекта.
- Двойной щелчок кнопкой мыши в любом свободном месте окна Quick-скрипта InTouch.
- Выполнение команды **Имя тэга (Tagname)** меню **Вставка (Insert)** в окне редактора Quick-скриптов InTouch.
- Нажатие клавиш ALT+N в окне редактора Quick-скриптов InTouch.
- Двойной щелчок кнопкой мыши пустого поля **Новое имя (New Name)** диалогового окна **Замена тэгов (Substitute Tagnames)**.
- Двойной щелчок кнопкой мыши на поле ввода **Tagname.FieldName** в диалоговом окне **Список переменных (Bind List Configuration)** программы SQL Access.



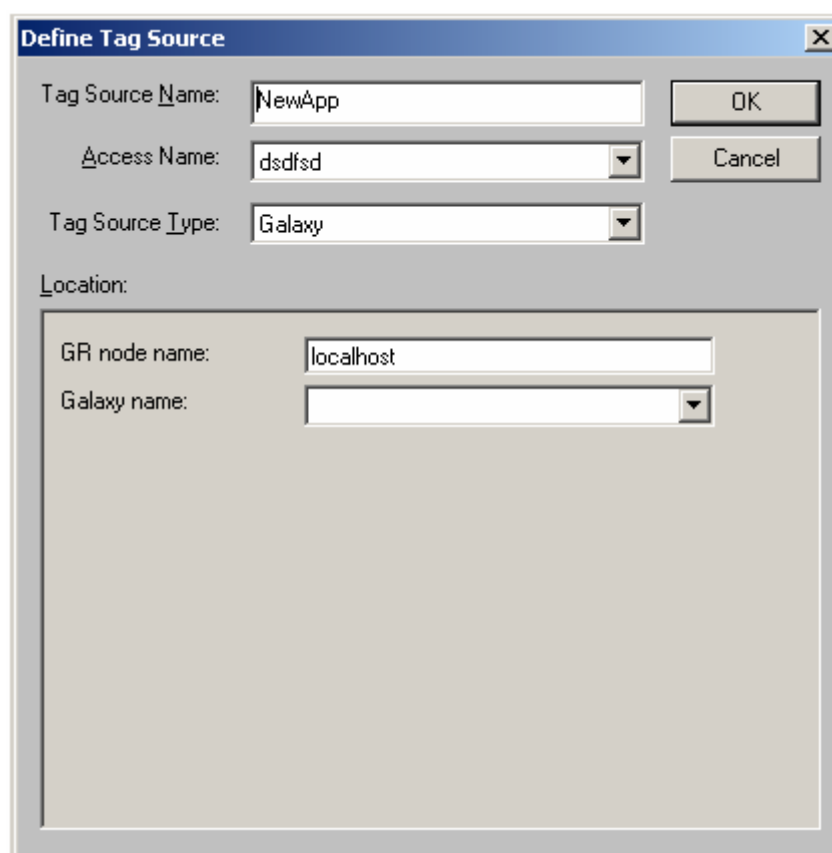
Подробнее об использовании браузера тэгов см. параграф "Браузер тэгов" главы 6 "Словарь переменных" в "Руководстве пользователя InTouch".

### Чтобы определить новый источник тэгов

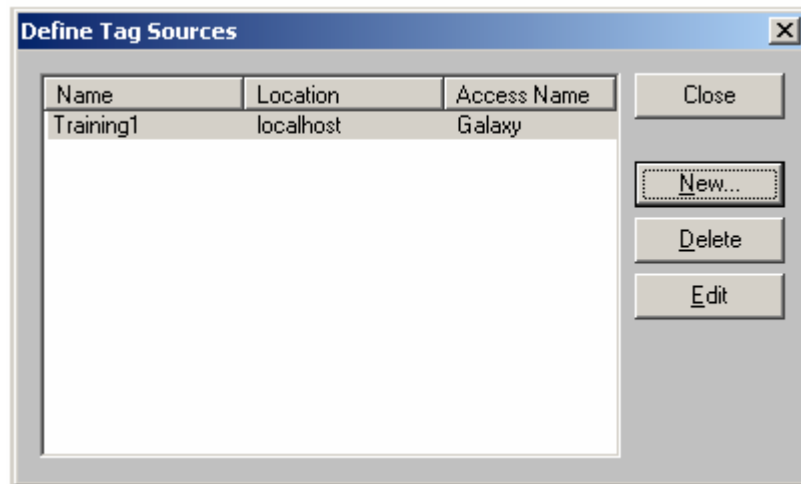
1. Щёлкните определения источников тэгов **Define Tag Source** (кнопка с многоточием справа от поля **Источник тэгов – Tag Source**). Появится окно **Определение источников тэгов (Define Tag Source)**.



2. Чтобы создать новый источник тэгов, щёлкните **Создать (New)**.



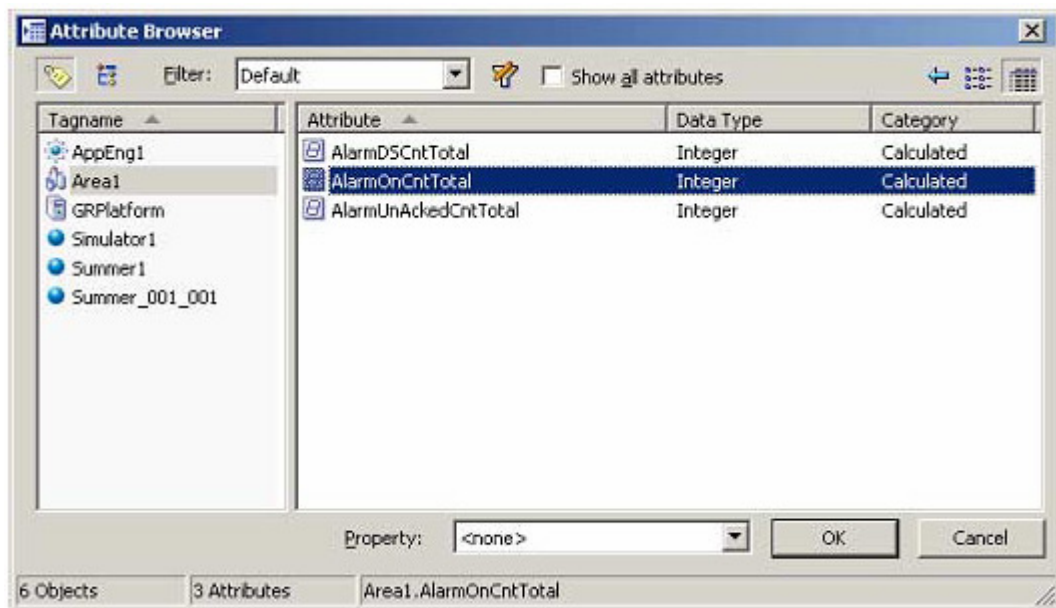
3. Введите в поле **Имя источника тэгов (Tag Source Name)** имя нового источника тэгов. Оно будет показано в списке **Источник тэгов (Tag Source)** в окне браузера тэгов (например Training1).
4. Выберите из списка **Имя канала доступа (Access Name)** и **Тип источника тэгов (Tag Source Type)** Galaxy.
5. Введите в поле **Имя узла Репозитория Galaxy (GR Node Name)** имя хост-компьютера, на котором находится Galaxy. Если она установлена на этом же компьютере, оставьте параметр по умолчанию (localhost).
6. Выберите имя системы из списка **Имя Galaxy (Galaxy Name)** (например, Training1). Окно **Определение источника тэгов (Define Tag Source)** будет выглядеть следующим образом:



7. Щёлкните **Закреть (Close)**. Появится окно браузера тэгов.

**Чтобы просмотреть ссылки на атрибуты в Galaxy**

1. Откройте окно браузера тэгов в режиме поиска без ограничений.
2. В поле **Источник тэгов (Tag Source)** выберите источник тэгов (например определённый ранее Training1). Появится сообщение о подключении к указанной Galaxy (Connecting to "training1"), после чего откроется окно браузера атрибутов.



3. Выберите объект и атрибут, которые должны быть использованы в приложении InTouch, и щёлкните **ОК**.

---

**Примечание.** При последующих попытках запустить браузер тэгов будет автоматически открываться окно браузера атрибутов. Чтобы переключиться в браузер тэгов, щёлкните кнопкой мыши изображение голубой стрелки в правом верхнем углу окна браузера атрибутов, ничего не выбирая до этого в его окне. Появится окно браузера тэгов с информацией из Словаря тэгов приложения InTouch.

---

Подробнее см. параграф "Просмотр объектов с помощью браузера атрибутов".



## ГЛАВА 9

# Контроль доступа

ArchestrA проверяет полномочия пользователей выполнять те или иные действия, включая следующие:

- Конфигурирование и манипулирование объектами в ИСР.
- Выполнение административных и служебных функций в консоли управления ArchestrA.
- Выполнение действий в рабочей системе.

Полномочиями определяются не только права доступа к пользовательским интерфейсам ArchestrA, но и доступ к атрибутам объектов и представляемым ими данным. Доступ к тому или иному атрибуту изображается соответствующим значком (см. главу "Редакторы объектов").

В каждой Galaxy, определённой в Репозитории Galaxy, имеется собственная модель контроля доступа. Она представляет собой трёхуровневую структуру конфигурирования, определяющую возможность создания и изменения следующих элементов:

- Групп прав доступа к конкретным объектам Galaxy.
- Пользовательских ролей, отображаемых на группы прав доступа, с определёнными правами администрирования и конфигурирования системы, а также выполнения требуемых действий в рабочем приложении.
- Пользователей, которые назначаются на определённые роли.

Подобная структура прав доступа представляет собой способ определения пользователей, которые назначаются на роли, для которых указываются соответствующие группы прав полномочий, необходимых для доступа к конкретным объектам Galaxy. Таким образом, права пользователя могут изменяться в зависимости от типа объекта, типа действия и типа технологического процесса.

Права доступа определяются в окне **Параметры доступа (Configure Security)**.

В настоящей главе приведено описание архитектуру контроля доступа ArchestrA и её применение для управления доступом пользователей к конфигурационным и рабочим параметрам IAS.

Подробнее о принципах контроля доступа в ArchestrA см. в Руководстве по разработке проектов FactorySuite A<sup>2</sup>.

## Содержание

- Определение прав доступа

## Определение прав доступа

Чтобы изменить параметры системы прав доступа в Galaxy, нужно соблюдать следующие условия:

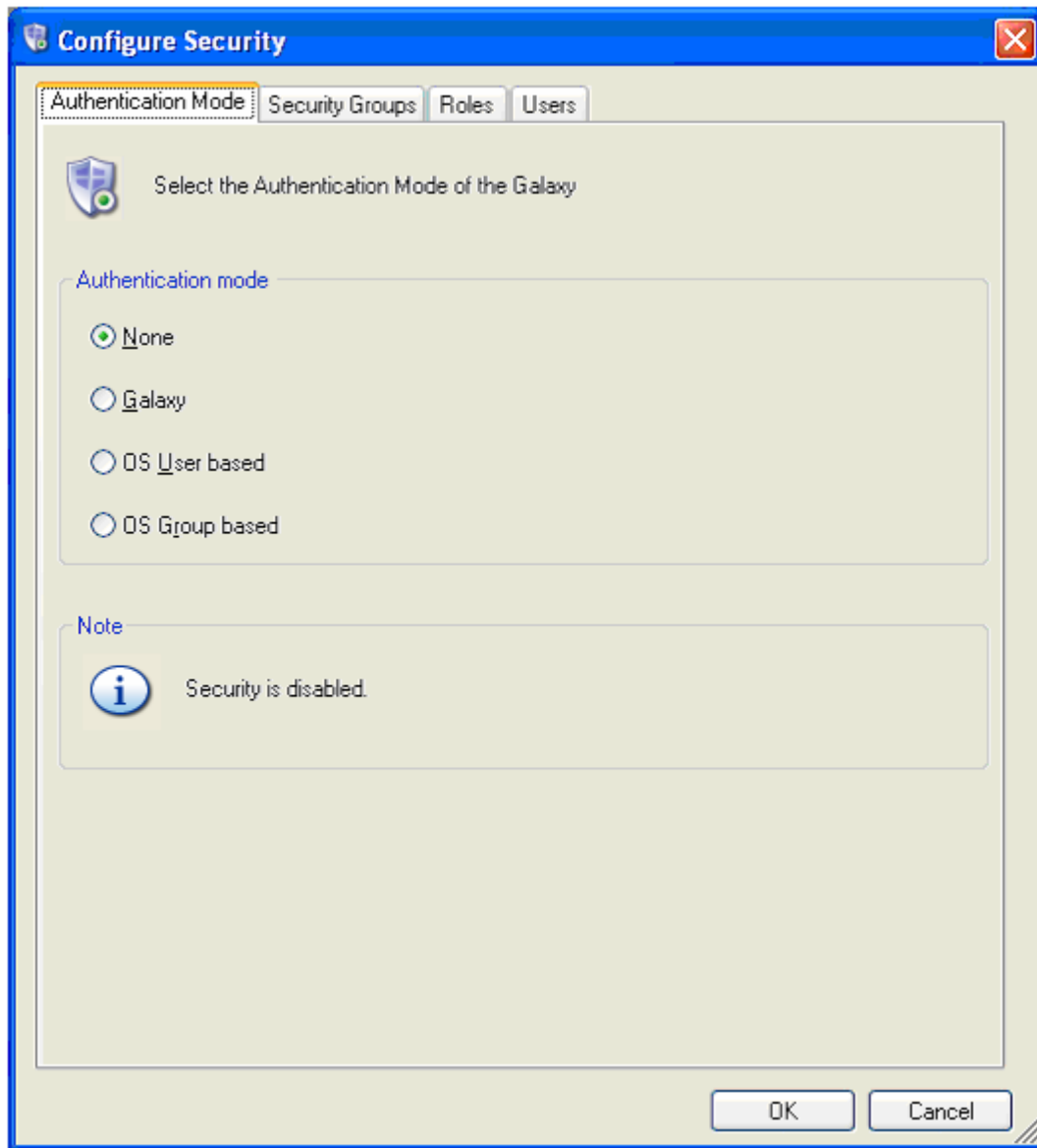
- К Galaxy не должен быть подключен ни один пользователь.
- Все объекты Galaxy должны быть освобождены пользователями (то есть должны быть не захваченными).
- Права пользователя, выполняющего действие, должны разрешать изменение конфигурационных параметров и параметров контроля доступа.

Если хотя бы одно из перечисленных условий не выполнено, при попытке запустить редактор прав доступа будет выдано сообщение об отказе. Во время изменения прав доступа ни один пользователь не сможет запустить ИСР, чтобы подключиться к Galaxy.

При выполнении команды **Проверка прав доступа (View Security)** вложенного меню **Конфигурировать (Configure)** меню **Galaxy** текущая архитектура прав доступа будет показана в режиме "только для чтения".

Чтобы изменить параметры доступа, поместите курсор мыши на пункт **Конфигурировать (Configure)** меню **Galaxy** и выполните команду **Права доступа (Security)** открывшегося меню. Появится окно **Параметры доступа (Configure Security)**.

## Способ идентификации пользователей



На странице **Режим идентификации (Authentication Mode)** определяется режим идентификации пользователей Galaxy:

- **Нет (None):** стандартный режим для всех новых Galaxy, называемый режимом открытого доступа (Open Security). В этом режиме любой пользователь может выполнять любые действия. Никакой проверки полномочий на конфигурирование или исполнение в Archestra не осуществляется. При запуске утилит IAS и рабочих процессов окна регистрации не отображаются.
- **Galaxy:** в этом режиме идентификация пользователя выполняется на основе сведений, имеющихся в локальной Galaxy. Этот режим выбирается для создания системы контроля доступа, контролируемого базой данных Galaxy.
- **Полномочия пользователя ОС (OS User Based):** в этом режиме права доступа определяются полномочиями пользователя в операционной системе. Этот режим выбирается, если идентификация должна выполняться для каждого пользователя отдельно при помощи механизма операционной системы (NT).

- **Полномочия группы ОС (OS Group Based):** в этом режиме права доступа определяются параметрами той группы пользователей операционной системы, в которую входит данный пользователь. При выборе этого режима становятся доступными следующие поля:
  - **Период регистрации (Login Time):** интервал времени в миллисекундах, в течение которого система определяет вхождение пользователя в одну из групп пользователей ОС, указанных в ролях ArchestrA. По истечении этого периода сведения извлекаются из локальной кэш-памяти. При успешной регистрации и длительности этого интервала, отличной от 0, сведения в локальной кэш-памяти обновляются. В случае превышения таймаута, если данный пользователь уже регистрировался ранее в системе ArchestrA с данного компьютера, нужные сведения берутся из локальной кэш-памяти. Если пользователь никогда не выполнял регистрацию с данного компьютера, процесс регистрации в системе ArchestrA завершается с ошибкой. Минимально допустимая длительность периода регистрации равна 0 мс, максимальная – 9999999 мс. Длительность по умолчанию равна 0 мс, что означает отключение данной функции (то есть таймаута регистрации нет). Этот параметр указывается, в основном, при работе в сетях с низкой скоростью передачи данных или с прерывистым доступом. Данная величина зависит как от скорости передачи данных в сети, так и от количества групп, определённых в ArchestrA. Иными словами, чем ниже скорость передачи данных в сети и чем больше количество групп, тем большим должно быть вводимое в это поле значение.
  - **Проверка ролей (Role Update):** интервал времени в миллисекундах между проверками принадлежности пользователя к каждой группе ОС при выполнении регистрации. С целью минимизации обращений к сети эта операция выполняется с частотой "одна роль в интервал". Минимально допустимое значение – 0 мс, максимально допустимое – 9999999. Длительность интервала по умолчанию равна 0 мс, что означает отключение данной функции (то есть пауза между проверками принадлежности пользователя и групп отсутствует). Этот параметр указывается, в основном, при работе в сетях с низкой скоростью передачи данных или с прерывистым доступом. Связи между этим параметром и периодом регистрации нет. Даже при наступлении таймаута регистрации проверка ролей будет осуществляться в фоновом режиме и, в конечном счёте, завершится обновлением в локальной кэш-памяти сведений о ролях, которые имеются у выполняющего регистрацию пользователя.

---

**Примечание.** При выборе того или иного метода идентификации следует обращать внимание на сообщения, выводимые в окно

**Примечание (Note).**

---

Установление того или иного режима идентификации приводит к следующим результатам:

- В режиме открытого доступа все пользователи имеют полномочия Defaultuser. В этом случае при изменении конфигурации системы, при выполнении административных функций и при запуске рабочего приложения отображение окна регистрации пользователя не производится. Окна регистрации пользователей открываются при установлении других режимов идентификации.
- После изменения режима идентификации разрешается работа только тех пользователей, сведения о которых были внесены во время конфигурирования нового режима. Сведения о других пользователях



не удаляются, но этим пользователям в новом режиме будет отказано в доступе.

- После установления режима идентификации (отличного от None) и закрытия описываемого окна нужно зарегистрироваться в системе. Это гарантирует ввод в действие нового режима идентификации. При нажатии в окне регистрации кнопки **Отмена (Cancel)** выполняется завершение работы ИСР.
- После изменения режима идентификации на **None** и нажатия кнопки **ОК** выполняется завершение работы ИСР.
- После установления режима идентификации в Galaxy каждый пользователь должен будет вводить в окне регистрации свои имя и пароль. Система контроля доступа сравнивает введённые сведения с информацией из базы данных Galaxy. Возможность выполнения действий в ИСР и в других компонентах ArchestrA определяется соответствующими данному пользователю ролями и полномочиями. Внешний вид графического интерфейса настраивается в соответствии с предпочтениями пользователя, определёнными в предыдущем сеансе (в частности, отображается та структура приложения, которая была в окне ИСР в момент его закрытия). Имя зарегистрировавшегося пользователя выводится в панели состояния ИСР.
- В режиме идентификации **Полномочия пользователя ОС (OS User Based)** пользователь в окне регистрации должен указывать свои имя, пароль и домен. Система контроля предоставляет пользователю операционной системы полномочия в ИСР.
- В режиме идентификации **Полномочия группы ОС (OS Group Based)** пользователь в окне регистрации также должен указывать свои имя, пароль и домен. Прежде всего система контроля проверяет полномочия пользователя для работы в ИСР, затем сравнивает мандат пользователя с параметрами группы операционной системы, отображёнными на роли в Galaxy.

---

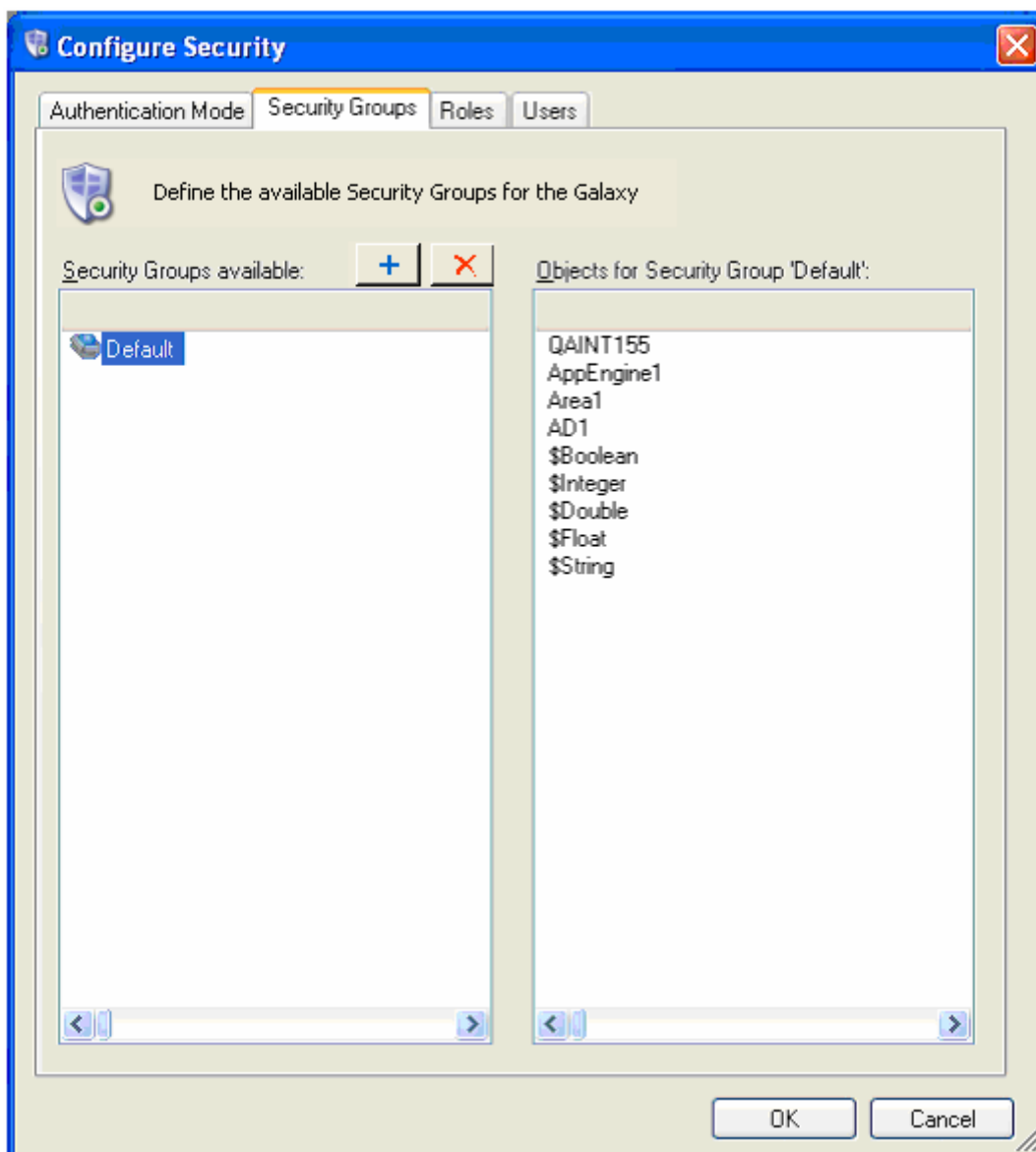
**Примечание.** В обоих режимах идентификации окно регистрации не открывается, если пользователь уже имеет права запуска данной утилиты ArchestrA.

---

- Для пользователя в системе контроля доступа может быть определено несколько наборов полномочий. Например, одна учётная запись может разрешать ему работать с объектами, но не с шаблонами экземпляров, другая, будучи учётной записью администратора, – работать с шаблонами и манипулировать списками пользователей ArchestrA. Чтобы переключаться между различными группами полномочий, оператор должен зарегистрироваться в системе как новый пользователь. Для этого нужно выполнить команду **Смена пользователя (Change User)** меню **Galaxy**.

Подробнее об идентификации пользователей на основе групп пользователей операционной системы см. параграф "Предоставление доступа в режиме проверки полномочий групп ОС".

## Группы прав доступа



Создание и настройка групп прав доступа выполняется на странице **Группы пользователей (Security Groups)**. Каждый объект Galaxy может быть связан только с одной группой прав доступа. На странице **Роли (Roles)** эти группы сопоставляются ролям пользователей. В данном случае (см. рисунок) все объекты Galaxy, перечисленные в правой панели, связаны с группой прав доступа по умолчанию Default.

Чтобы создать новую группу прав доступа, щёлкните символ "+" и введите её имя в появившемся окне **Имеющиеся группы прав доступа (Security Groups Available)**. Определение объектов, которые будут связаны с новой группой, выполняется путём перетаскивания их с помощью мыши из списков уже существующих групп в список новой группы.

---

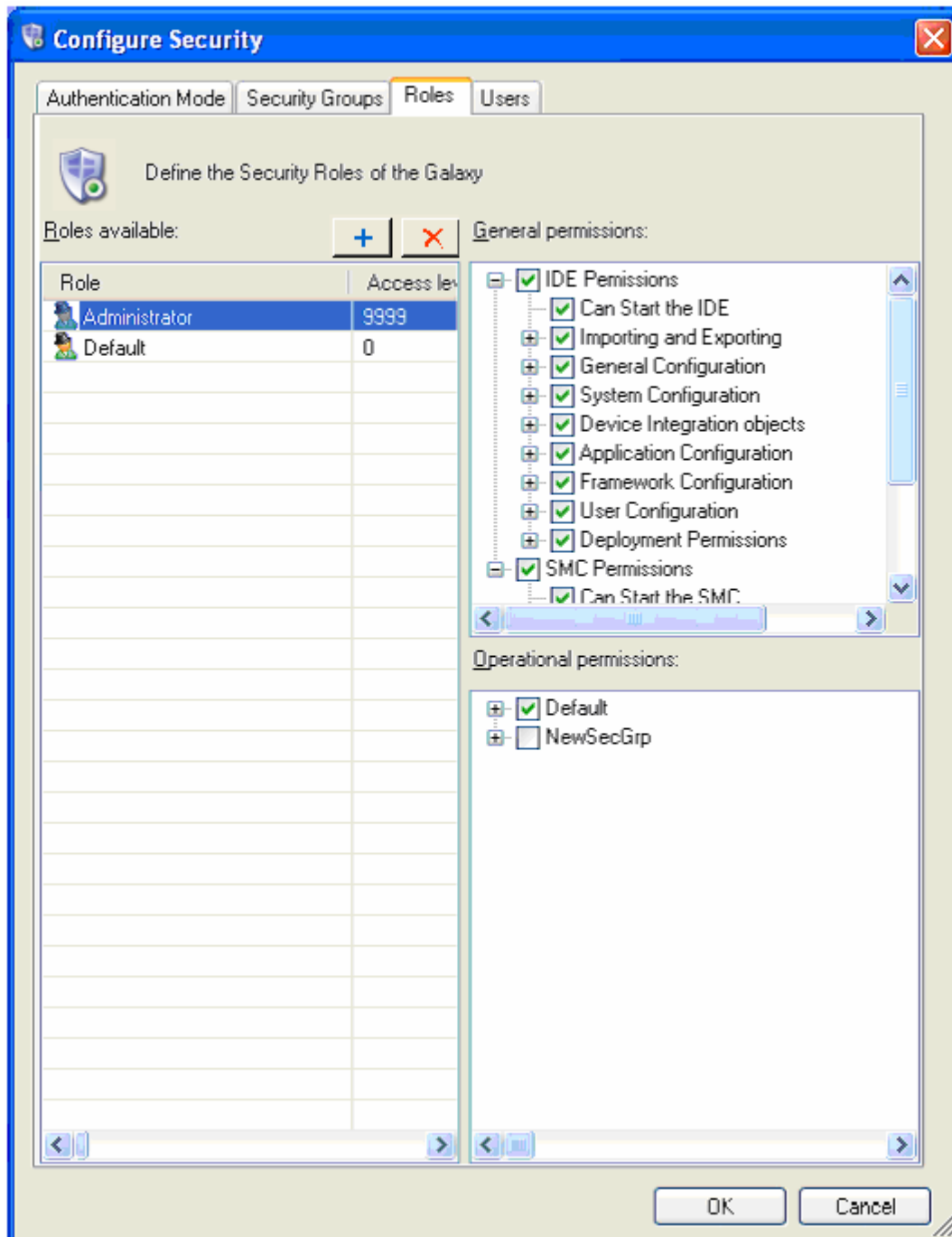
**Примечание.** Регистр букв при указании имён групп прав доступа не важен (то есть создать два набора, имена которых различаются только регистром букв, невозможно).

---

Чтобы удалить группу прав доступа, нужно выделить её имя и нажать кнопку с символом "X". Если с группой связаны объекты AutomationObject, удалить её будет невозможно. Кроме того, нельзя удалить группу прав доступа Default.

Подробнее об именовании групп прав доступа см. параграф "Допустимые имена и символы".

## Роли



Роли пользователей, которые могут выполнять те или иные действия в системе, определяются на странице **Роли (Roles)**. По умолчанию в системе уже определены две роли: **Администратор (Administrator)** и **Default (Стандартный пользователь)**.

Чтобы увидеть полномочия, выделите роль в расположенных справа списках **Общие полномочия (General Permissions)** или **Полномочия времени исполнения (Operational Permissions)**. В панели общих полномочий перечислены действия, которые разрешаются во время конфигурирования приложения и выполнения административных задач, в панели полномочий времени исполнения перечислены группы прав

доступа, определённые на странице **Группы прав доступа (Security Groups)**. По умолчанию пользователи с ролью администратора обладают всеми полномочиями в системе.

---

**Примечание.** Изменить состав общих полномочий, определённых для роли администратора, невозможно.

---

Чтобы определить новую роль, щёлкните символ "+" и введите её название в окне **Имеющиеся роли (Roles Available)**. Для новых ролей нужно определять состав общих полномочий и полномочий времени исполнения.

Чтобы удалить роль, выделите её название в списке и щёлкните символ "X". Если эта роль присвоена каким-либо пользователям, удалить её нельзя. Кроме того, невозможно удалить роли стандартного пользователя и администратора, причём сократить перечень полномочий администратора также нельзя.

Подробнее об именовании ролей см. параграф "Допустимые имена и символы".

**AccessLevel** представляет собой параметр InTouch™, который используется для определения приоритетов выполняемых функций. В модели контроля доступа сервера IAS он соответствует кодам уровней доступа InTouch, но при этом никак не связан с приоритетами выполнения. Максимальное значение этого поля – 9999, минимальное – 0. Если пользователю будет назначено несколько ролей с различными кодами уровня доступа, в InTouch будет передано больший из них.

---

**Примечание.** Если название роли выделено полужирным шрифтом красного цвета, она для выбранного режима идентификации является недопустимой.

---

## Общие полномочия

Для новой роли могут быть определены следующие общие полномочия:

- Запуск ИСР (Can Start an IDE).
- Импорт и экспорт (Importing and Exporting).
  - Может импортировать (Can Import).
  - Может выполнять загрузку/выгрузку Galaxy (Can Utilize Galaxy Load/Galaxy Dump).
  - Может экспортировать (Can Export).
- Стандартное конфигурирование (General Configuration):
  - Может изменять используемые экземпляры (Can Modify Deployed Instances) – эти полномочия представляют собой супернабор всех типов полномочий, включая группы System Configuration, Device Integration Objects и Application Configuration.

---

**Внимание!** Если для роли указаны полномочия "Can Modify Deployed Instances", нужно также указать соответствующие полномочия на создание, изменение и удаление объектов в группах System Configuration, Device Integration Objects и Application Configuration, для того чтобы пользователь мог захватывать объекты и отменять захват.

---

- Может отменять ввод комментариев во время изменения (Can disable change comments).
- Может игнорировать захват (Can override checkout).
- Может выгружать рабочие параметры (Can upload from runtime).

- Конфигурирование системы (System Configuration):
  - Может создавать, изменять и удалять шаблоны системных объектов – платформ и приложений (Can Create/Modify/Delete System Object Templates – Platforms and Engines).
  - Может создавать, изменять и удалять экземпляры системных объектов – платформ и приложений (Can Create/Modify/Delete System Object Instances – Platforms and Engines).
  - Может создавать, изменять и удалять зоны (Can Create/Modify/Delete Area Objects).
- Объекты интеграции устройств (Device Integration Objects):
  - Может создавать, изменять и удалять шаблоны объектов интеграции устройств (Can Create/Modify/Delete Device Integration Object Templates).
  - Может создавать, изменять и удалять экземпляры объектов интеграции устройств (Can Create/Modify/Delete Device Integration Object Instances).
- Конфигурирование приложений (Application Configuration):
  - Может создавать, изменять и удалять шаблоны объектов ApplicationObject (Can Create/Modify/Delete Application Object Templates).
  - Может создавать, изменять и удалять экземпляры объектов ApplicationObject (Can Create/Modify/Delete Application Object Instances).
- Конфигурирование Framework (Framework Configuration):
  - Может изменять структуры прав доступа (Can Modify the Security Model).
  - Может изменять региональных параметров (Can Modify Localization Settings).
  - Модификация параметров временной синхронизации (Can Modify the Time synchronization settings).
- Конфигурирование пользователей (User Configuration):
  - Может создавать, изменять и удалять сведения о пользователях (Can Create/Modify/Delete users).
  - Может изменять сведения о себе (Can Modify own User Information).
- Разрешение рассылки (Deployment Permissions):
  - Может рассылать/отменять использование системных объектов (Can Deploy/Undeploy System Objects).
  - Может рассылать/отменять использование объектов зоны (Can Deploy/Undeploy Area Objects).
  - Может рассылать/отменять использование объектов приложения (Can Deploy/Undeploy Application Objects).
  - Может рассылать/отменять использование объектов DeviceIntegration (Can Deploy/Undeploy DeviceIntegration Objects).
  - Может помечать объект как неиспользуемый (Can Mark an Object as Undeployed).
- Может запускать SMC (Can Start the SMC):

- Может запуск/останавливать платформы/приложения (Can Start/Stop Engine/Platform).
- Может записывать данные в объекты gObject с помощью Object Viewer (Can write to gObject Attributes using Object Viewer).

## Полномочия времени исполнения

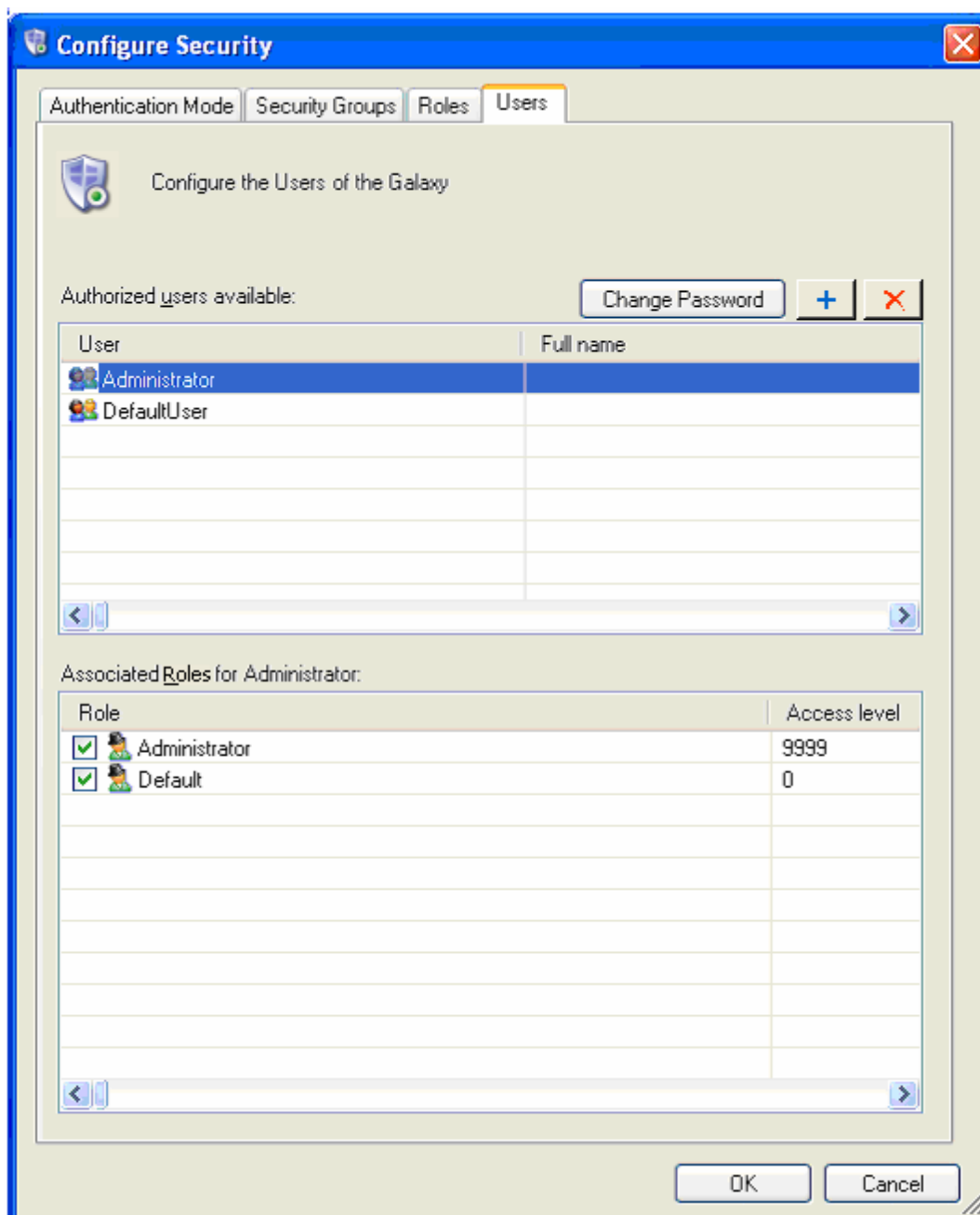
Для каждой роли могут быть определены следующие полномочия времени исполнения рабочего приложения:

- Может изменять атрибуты "Использование" (Can Modify "Operate" Attributes): пользователи могут выполнять определённые стандартные операции, в число которых входит установка значений атрибутов ПИД-объекта типа Setpoint (Уставка), Output (Выход) и Control Mode (Режим управления), управление объектами Discrete Device (Логическое устройство) и др.
- Может изменять атрибуты "Настройка" (Can Modify "Tune" Attributes): конечные пользователи могут настраивать значения атрибутов в рабочей системе. Примеры подобной настройки: изменение контрольных точек алармов и чувствительности ПИД-регуляторов.
- Может изменять атрибуты "Конфигурирование" (Can Modify "Configure" Attributes): пользователи могут изменять значения атрибутов. Нужно, чтобы сначала они отменяли сканирование объекта. Установка значения этого атрибута рассматривается как одно из существенных изменений конфигурации (например, смена регистра ПЛК, передающего данные на вход объекта Логическое устройство – Discrete Device).

Если в качестве режима идентификации указан режим **Полномочия группы ОС (OS Group Based)**, поиск доменов и групп пользователей в доменах можно осуществлять с помощью браузера.

После определения прав доступа к объектам, полномочий для ролей и полномочий времени использования можно назначать пользователям роли.

## Пользователи



На странице **Пользователи (Users)** отображаются сведения о пользователях Galaxy и о том, какие роли им назначены. Чтобы назначить пользователю определённую роль, нужно выделить его имя в панели **Уполномоченные пользователи (Authorized Users Available)** и затем выделить соответствующую роль в списке **Соответствующие роли для <имя пользователя> (Associated Roles for <user name>)**. Пользователям могут быть назначены более одной роли.

После выбора режима идентификации **OS Group Based** эта страница открывается только в режиме просмотра. В момент регистрации пользователи автоматически добавляются в список **Уполномоченные пользователи (Authorized Users Available)**.

**Внимание!** При указании любого из режимов идентификации на основе механизмов ОС сведения о пользователях с локальными учётными записями добавляются в список в следующем формате:  
.\<имя\_пользователя>. Во время просмотра в InTouch сведений об алармах

и событиях IAS символ точки заменятся на имя домена пользователя. При указании режима идентификации **Полномочия группы ОС (OS Group Based)** нужно, чтобы сведения об этих учётных записях существовали на всех узлах Galaxy, для того чтобы регистрация пользователя в системе завершилась успешно на любом компьютере сети.

При создании новой Galaxy в ней автоматически определяются два типа пользователей: Администратор (Administrator) и Стандартный пользователь (DefaultUser). В режиме открытого доступа их удалить невозможно. По умолчанию им назначаются роли соответственно Administrator и Default.

**Примечание.** Имена пользователей и названия ролей, выделенные красным цветом, в данном режиме идентификации являются недействительными.

Чтобы создать запись о новом пользователе, щёлкните символ "+" и введите его имя в списке уполномоченных пользователей. Указанное имя должно быть уникальным в Galaxy. Чтобы удалить сведения о пользователе, выделите в списке его имя и щёлкните символ "X".

**Примечание.** Удалить сведения о пользователе, который в текущий момент зарегистрирован в системе, нельзя.

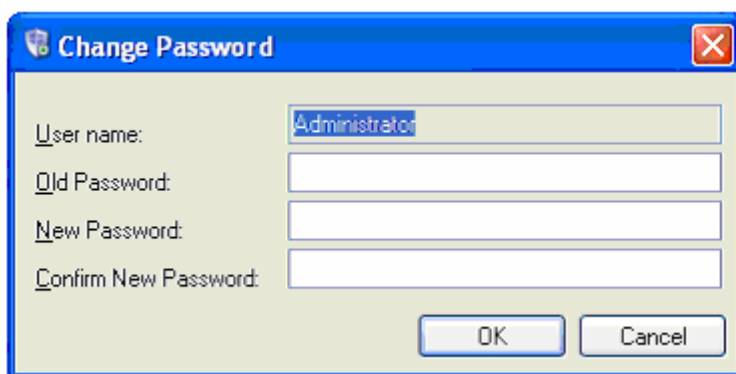
Подробнее о правилах именования см. параграф "Допустимые имена и символы".

По умолчанию, только что определённому пользователю назначается Стандартная (Default) роль (это назначение изменить нельзя, т.к. у каждого пользователя имеется данная роль), но не роль администратора. Чтобы изменить назначение, дважды щёлкните текстовое поле и введите нужную информацию.

**Примечание.** По умолчанию, все пользователи в дополнение к определённому для них ролям получают роль Default. Это значит, что, если пользователи должны иметь полномочия, определяемые новой для них, а не стандартной ролью, полномочия стандартной роли нужно изменить. В противном случае все пользователи будут иметь более широкий круг полномочий, которые определяются для стандартной роли Default во время установки системы.

Определить для каждого пользователя пароль можно, нажав кнопку **Изменить пароль (Change Password)**. Появится окно **Изменение пароля (Change Password)**.

**Внимание!** При установлении режима идентификации на основе механизмов ОС изменение пароля в этом окне приводит к изменению пароля пользователя в операционной системе.



Введите данные, требуемые для идентификации пользователя в среде конфигурирования, администрирования и исполнения рабочей программы.



---

**Примечание.** Пользователь может зарегистрироваться как администратор в любом режиме идентификации, за исключением ситуаций, когда контроль доступа отменён. Зарегистрировавшись в узле Репозитория Galaxy как администратор, пользователь может изменить пароль любого пользователя Galaxy без указания его предыдущего пароля, в поле **Старый пароль (Old Password)**. В режиме идентификации Galaxy можно также изменить **Имя пользователя (User name)**. В режиме идентификации на основе механизмов ОС в этом поле отображается имя пользователя в операционной системе. При этом изменить его невозможно.

---

Введя нужные данные, щёлкните **ОК** для ввода их в действие или кнопку **Отмена (Cancel)** для возврата к предыдущим параметрам контроля доступа.

## После изменения прав доступа

При изменении прав доступа нужно учитывать следующие обстоятельства:

- После изменения прав доступа осуществляется перезапуск ИСР.
- Объекты, для которых была указана иная группа прав доступа, помечаются как "pending update" (ожидающие обновления) и должны быть повторно отосланы в место использования, для того чтобы изменения прав доступа вступили в действие.
- В режимах идентификации на основе механизмов ОС изменение параметров доступа приводит к обновлению списков пользователей, их полных имён и групп ОС, если какие-либо сведения изменились также в операционной системе.

## Предоставление доступа в режиме проверки полномочий групп ОС

Выбор режима идентификации **Полномочия группы ОС (OS Group Based)** должен основываться на глубоком понимании принципов функционирования ОС Windows в области полномочий пользователей, групп пользователей и контроля доступа. Режим идентификации в ArchestrA опирается на эти функции Windows.

Нужно учитывать следующие уникальные для режима **Полномочия группы ОС (OS Group Based)** моменты:

- Недавно определённый пользователь, работающий на компьютере, который не имеет доступа к узлу Galaxy, не может устанавливать значения атрибутов объекта на удалённом узле, если он до этого ещё на этом узле ни разу не регистрировался (даже если у этого пользователя достаточно полномочий для выполнения операции во время работы приложения). Чтобы иметь возможность установки значений атрибутов на удалённом узле, нужно хотя бы раз на нём зарегистрироваться.
- Если после регистрации в ArchestrA из рабочей станции, принадлежащей домену А, контроллер этого домена прекращает функционирование, последующие регистрации выполняются на основе данных, сохранённых в локальной кэш-памяти. После того как работоспособность контроллера домена будет восстановлена, во время восстановления его каналов связи зарегистрироваться в системе будет невозможно. Если в период неработоспособности контроллера имя и пароль пользователя изменяются, возможность локальной работы будет обеспечена старыми сведениями. Чтобы выполнять операции на удалённых узлах, нужно попытаться зарегистрироваться в системе с использованием новых регистрационных данных. Неуспешная

регистрация означает, что контроллер восстанавливает свои каналы связи, и попытку регистрации следует повторить позже.

- Попытки регистрации в ArchestrA из рабочей станции с ОС Windows 2000 будут заканчиваться неудачей до тех пор, пока в политиках локальной безопасности не будет должным образом определена привилегия TCB. Для этого нужно выполнить следующее: на компьютере с ОС Windows 2000 Server нажать кнопку **Пуск (Start)** Панели задач, поместить указатель над пунктом **Программы (Programs)**, затем – над пунктом **Администрирование (Administrative Tools)** и щёлкнуть пункт **Политики локальной безопасности (Local Security Policies)**; на компьютере с операционной системой Windows 2000 Professional нажать кнопку **Пуск (Start)** Панели задач, щёлкнуть пункт **Панель управления (Control Panel)**, затем – пункт **Администрирование (Administrative Tools)**, после чего – пункт **Параметры локальной безопасности (Local Security Settings)**. В левой панели окна параметров локальной безопасности раскройте папку **Локальные политики (Local Policies)**, после чего раскройте папку **Назначение прав пользователя (User Rights Assignment)**. Дважды щёлкните пункт **Работа как части операционной системы (Act as a part of operating system)**. В появившемся окне **Параметр локальной политики безопасности (Local Security Policy Setting)** добавьте пользователя (зарегистрировавшегося на компьютере) путём установки флажка **Параметр локальной политики (Local Policy Setting)** и щёлкните **ОК**. Для ввода в действие новой привилегии TCB нужно завершить сеанс работы системы и зарегистрироваться в ней повторно. Чтобы определить привилегию TCB, нужно являться администратором системы.
- Изображение списка доменов и групп пользователей в браузере групп определяется тем, сконфигурирован ли домен пользователя как Mixed (смешанный) или Native (чистый). Изображение пользователя должно соответствовать списку доменов, которое выводится в окно административного средства Windows манипулирования параметрами домена. Для определения параметров безопасности группа доменов, сконфигурированная как "Distribution" (Распределение), а не "Security" (Защищённые), не подходит.
- Полное имя пользователя будет недоступно клиентам (в частности, не будет отображаться в окне приложения InTouch), если контроллер домена отключен от сети во время первой регистрации пользователя в ArchestrA. Если пользователь регистрировался ранее в ArchestrA, когда контроллер был подключен к сети, его полное имя будет взято из кэш-памяти и передано клиенту, даже если контроллер домена был отключен от сети во время последующей регистрации этого пользователя в ArchestrA.

# Управление Репозиторием Galaxy

Репозиторий Galaxy представляет собой центральную базу данных, с помощью которой осуществляется управление всеми приложениями IAS. Каждое приложение представлено в Репозитории отдельной Galaxy.

В настоящей главе содержится описание административных средств управления Репозиторием и содержащимися в нём системами Galaxy.

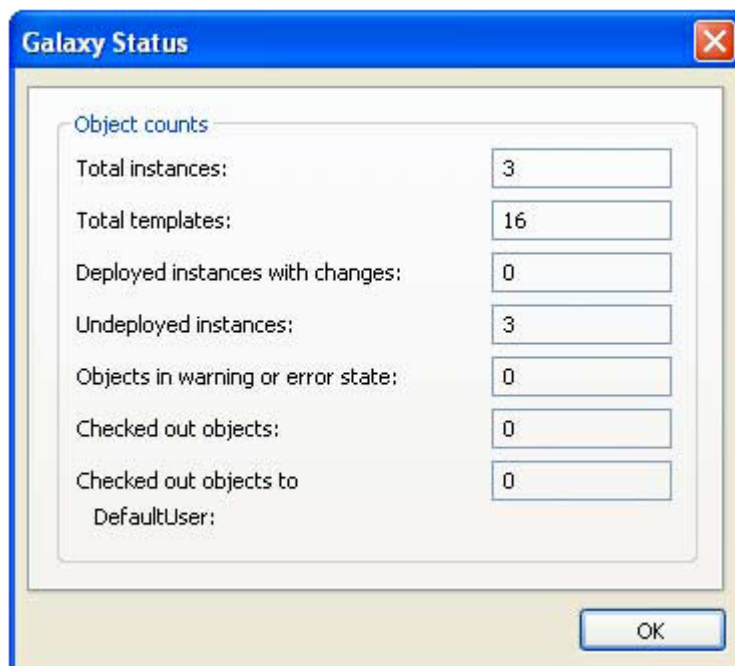
## Содержание

- Определение состояния Galaxy
- Создание новой Galaxy
- Наличие нескольких Galaxy в одном Репозитории Galaxy
- Загрузка и выгрузка Galaxy
- Выгрузка рабочих параметров в Galaxy
- Резервное копирование и восстановление Galaxy
- Конфигурирование главного источника показаний времени

## Определение состояния Galaxy

**Чтобы определить состояние Galaxy в Репозитории**

1. Подключитесь к Galaxy.
2. Выполните команду **Состояние Galaxy (Galaxy Status)** меню Galaxy. Появится окно **Состояние Galaxy (Galaxy Status)**.

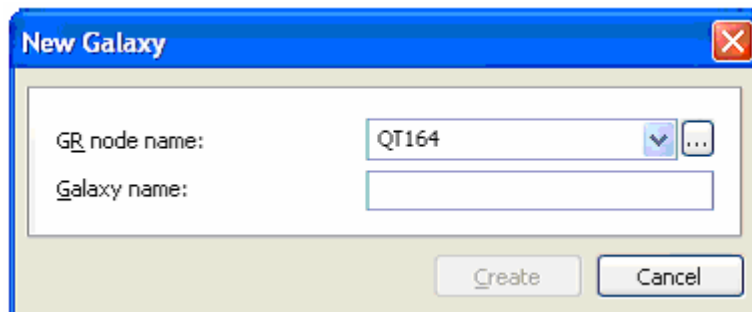


Информация, выводимая в это окно, может использоваться как общая характеристика системы, а также как отправная точка для её отладки. Особое значение в этом случае имеет отображаемое количество объектов в состоянии ошибки или предупреждения.

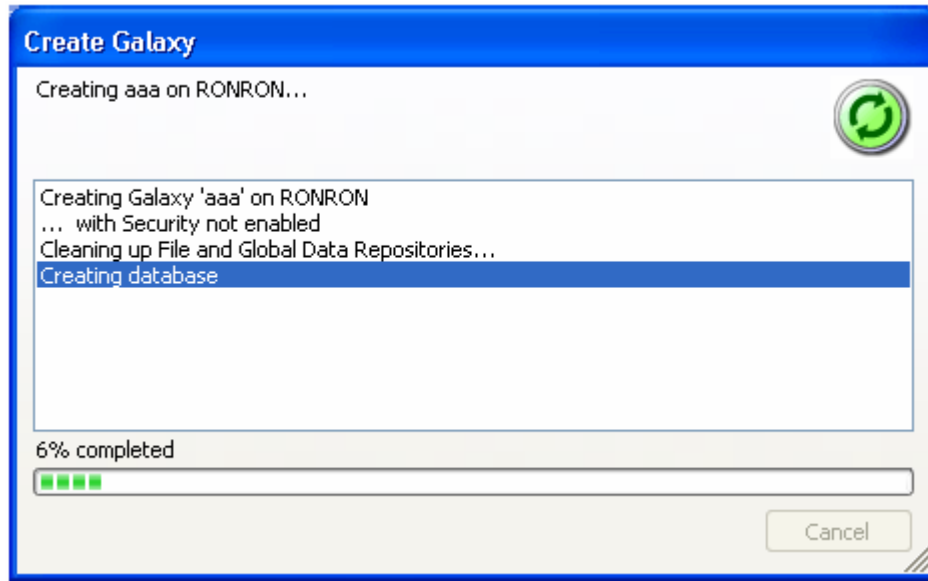
## Создание новой Galaxy

Чтобы создать новую Galaxy

1. Щёлкните **Новая Galaxy (New Galaxy)** в окне **Соединение с Galaxy (Connect to Galaxy)**. Появится окно **Новая Galaxy (New Galaxy)**.



2. Выберите из списка **Имя узла Репозитория Galaxy (GR Node Name)** компьютер, на котором должна быть создана новая Galaxy.
3. Введите в поле **Имя Galaxy (Galaxy Name)** имя создаваемой Galaxy.
4. Для запуска процесса создания новой Galaxy щёлкните **Создать (Create)**. Появится окно **Создание Galaxy (Create Galaxy)**, в котором будет отображаться ход выполнения команды. Чтобы вернуться в окно **Соединение с Galaxy (Connect to Galaxy)**, не создавая новую систему, щёлкните **Отмена (Cancel)**.



---

**Примечание.** Все новые системы создаются с открытым доступом. Сведения об определении прав доступа см. в предыдущей главе.

---

Щёлкните **Заккрыть (Close)**, чтобы вернуться в окно соединения с Galaxy, и затем кнопку **Отмена (Cancel)**, чтобы закрыть его.

## Наличие нескольких Galaxy в одном Репозитории Galaxy

В одном и том же Репозитории Galaxy может быть создано одновременно несколько Galaxy. Сконфигурировав одну систему, можно в ИСП переключиться на другую и продолжить конфигурировать её.

Вместе с тем, загружать Galaxy на компьютеры сети из Репозитория Galaxy можно только по одной Galaxy за один раз.

Допускается рассылать объекты одной системы, отменять их использование и затем рассылать объекты другой. Однако попытка переслать объекты одной Galaxy на компьютер, на котором продолжают использоваться объекты другой системы, завершится неудачей.

## Загрузка и выгрузка Galaxy

Объекты и их конфигурационные параметры могут быть экспортированы в виде файла в формате CSV (Comma Separated Values – разделённые запятыми значения) и затем импортированы в другую систему. Более подробно об экспортировании и импортировании информации из БД Galaxy см. главу "Объекты".

## Выгрузка рабочих параметров в Galaxy

Изменения характеристик некоторых атрибутов (Writeable\_UC, Writeable\_UC\_Lockable, Writeable\_USC, Writeable\_USC\_Lockable), выполненные в среде конфигурирования, могут быть реализованы в среде исполнения с некоторой задержкой. В результате значения этих атрибутов некоторое время в среде конфигурирования и среде исполнения могут быть различными.

Изменённые в рабочей конфигурации параметры могут быть пересланы в базу данных Galaxy, что обеспечит их идентичность в обоих случаях.

Таким образом, при повторной рассылке объектов в места использования или при отмене их использования с повторной рассылкой в будущем сделанные изменения будут сохранены.

#### Чтобы загрузить характеристики рабочей среды в БД Galaxy

1. Выделите соответствующий объект в общей структуре или структуре использования. Можно выделить несколько объектов, нажав одновременно со щелчком кнопки мыши клавишу SHIFT или CTRL.
2. Выполните команду **Загрузка рабочих характеристик (Upload Runtime Changes)** меню **Объект (Object)**. Характеристики атрибутов объекта в рабочей среде будут сохранены в БД Galaxy.

При попытке выделить объект, захваченный этим же пользователем, будет выдано соответствующее предупреждение. Если пользователь нажмёт кнопку продолжения, все изменения в конфигурации объекта, выполненные к текущему моменту, будут утеряны.

---

**Примечание.** Копирование рабочих характеристик объектов, захваченных другими пользователями, не выполняется.

---

После копирования рабочих характеристик объекты получают новый номер версии. Кроме того, в журнале изменений создаются соответствующие записи. Объекты в рабочей системе также получают новый номер версии, идентичный записанному в базе данных.

## Резервное копирование и восстановление Galaxy

Создавать резервные копии Galaxy нужно. Это позволяет избежать катастрофических последствий в случае отказов компьютеров или хакерских атак на систему.

Резервные копии создаются с помощью менеджера БД Galaxy DataBase Manager, который входит в состав утилит системной консоли управления (System Management Console) ArchestrA. Чтобы запустить приложение менеджера, щёлкните **Пуск (Start)** в Панели задач, поместите курсор мыши на пункты **Программы (Programs)** и **Wonderware** и затем щёлкните пункт **System Системная консоль управления (Management Console)**.

Подробнее см. документацию по использованию менеджера БД Galaxy DataBase Manager.

## Конфигурирование главного источника показаний времени

Главный источник показаний времени представляет собой сервер сетевого времени, с часами которого могут синхронизироваться все остальные узлы сети. Им может быть как узел Galaxy, так и любой другой не входящий в ArchestrA узел. Узлы ArchestrA в Galaxy периодически синхронизируют свои часы с показаниями этого источника.

Если источник показаний времени или клиент работают под управлением ОС Windows 2003 Server или Windows XP, перед использованием этой функции нужно установить на компьютере пакет исправлений Microsoft (в соответствии со следующей таблицей).

Операционная система	Пакет исправлений
Windows XP	WindowsXP-KB823456-x86-ENU.exe

Windows 2003 Sever	WindowsServer2003-KB823456-x86-ENU.exe
--------------------	--

Эти пакеты можно получить в подразделении Microsoft по поддержке продукции компании.

Если источник показаний времени не является узлом Galaxy (независимо от типа операционной системы), дополнительно нужно изменить значения некоторых ключей реестра. Получить необходимые сведения и файлы установки значений ключей реестра можно в технической службе поддержки Wonderware.

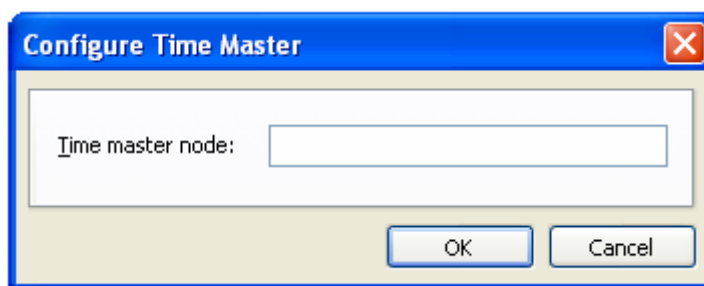
---

**Внимание!** Обновление программного обеспечения и значений ключей реестра нужно выполнять до конфигурирования узла в качестве источника показаний времени.

---

### Чтобы сконфигурировать узел как главный источник показаний времени

1. Выполните предварительно требуемое обновление программного обеспечения и значений ключей реестра.
2. Поместите курсор мыши на пункт **Конфигурировать (Configure)** меню **Galaxy** и выполните в открывшемся меню команду **Источник показаний времени (Time Master)**. Появится окно **Конфигурирование источника показаний времени (Configure Time Master)**.



3. Введите в поле **Узел источника показаний времени (Time Master Node)** полное сетевое имя требуемого узла.
4. Щёлкните **ОК** для подтверждения или кнопку **Отмена (Cancel)** для отмены операции.

Часы заданного узла будут служить в качестве главного источника временных отметок всех функций. В ArchestrA нет собственного алгоритма синхронизации времени. Синхронизация осуществляется средствами службы времени Windows. По умолчанию синхронизация часов выполняется сначала один раз в 45 минут, а после трёх успешных попыток синхронизация – один раз в восемь часов.

Платформы WinPlatform начинают синхронизировать часы своего узла при их пересылке.

Источник показаний времени может находиться в другом часовом поясе. Показания времени на каждом узле IAS в этом случае будут соответствовать часовому поясу, в котором эти узлы находятся.

---

**Внимание!** Если узлы ArchestrA Galaxy уже являются членами домена Windows 2000, администратор этого домена, возможно, уже выполнил синхронизацию времени. В этом случае конфигурирование главного источника показаний времени для Galaxy является необязательным и может противоречить уже установленной схеме синхронизации.

Кроме того, синхронизация времени важна, если один или несколько узлов системы функционируют под управлением ОС Windows XP. Эта система поддерживает протокол сетевой идентификации, который требует синхронизации часов узлов, для того чтобы они могли взаимодействовать

между собой. Обмен данными между узлами выполняться не будет, если расхождение показаний их часов будет превышать некоторую установленную величину (например пять минут). Это означает, что, если часы узла Galaxy с ОС Windows XP будут идти вперед или отставать на указанную величину временного допуска, операции ArchestrA, такие как рассылки объектов, выполняться не будут.

---



# Резервирование компонентов ArchestrA

Резервирование представляет собой дублирование в системе её важнейших компонентов. Основная цель резервирования заключается в обеспечении безотказной работы системы в случае отказа какого-либо её элемента.

Инфраструктура ArchestrA поддерживает два типа резервирования: создание пар объектов AppEngine на случай отказа компьютера или программных средств и конфигурирование резервных каналов связи с одним или несколькими ПЛК.

Аварийное переключение представляет собой передачу управления резервному компоненту без остановки системы. Аварийное переключение может происходить как в результате сбоя, так и в результате команды оператора (принудительное переключение).

Подробнее о резервировании см. Руководство по разработке проектов FactorySuite A<sup>2</sup>.

## Содержание

- Общие сведения
- Резервирование объектов AppEngine
- Резервирование каналов связи

## Общие сведения

ArchestrA поддерживает резервирование двух основных элементов: объектов AppEngine и каналов сбора данных. В обоих случаях в параметрах этих компонентах должно быть указано наличие резервирующих составляющих. Резервирование объектов AppEngine и каналов связи является взаимоисключающим.

При резервировании объектов AppEngine нужно конфигурировать как сам объект, так и две WinPlatform. В случае резервирования каналов связи нужно конфигурировать два объекта DIObject (источники данных) и объект RedundantDIObject.

В следующих параграфах содержатся следующие сведения:

- Способы конфигурирования пар объектов AppEngine/WinPlatform и источников данных RedundantDIObject/DIObject.
- Внедрение объектов, связанных с резервированием, и выполнение других операций в ArchestrA.
- Описание действий, выполняемых системой в случае отказа одного из компонентов.

## Термины

В контексте резервирования в руководстве используются следующие термины:

### В среде конфигурирования:

- **Главный (primary) объект:** объект, который играет основную, или центральную роль в реализации прикладных функций во время работы приложения. Это либо объект AppEngine, который резервируется, либо объект DIObject, который используется первым в качестве источника данных для приложения.
- **Резервный (backup) объект:** объект, который обеспечивает выполнение прикладных функций при прекращении функционирования главного объекта. Это либо объект AppEngine, создаваемый инфраструктурой ArchestrA при разрешении резервирования главного объекта AppEngine, либо объект DIObject, который не должен использоваться первым в качестве источника данных для приложения.

### В среде исполнения:

- **Активный (active) объект:** объект, который в текущий момент обеспечивает выполнение прикладных функций. Это либо объект AppEngine, который объединяет объекты ApplicationObject и обеспечивает их функционирование, либо объект DIObject, которые обеспечивает поступление технологических данных посредством объекта RedundantDIObject.
- **Пассивный (standby) объект:** пассивный объект, ожидающий передачи управления в результате сбоя активного объекта или принудительного переключения. Это либо объект AppEngine, контролирующий состояние активного объекта AppEngine, либо объект DIObject, который не обеспечивает поступление технологических данных посредством объекта RedundantDIObject.

Главный/резервный и активный/пассивный объекты образуют дублированные пары. В случае объектов AppEngine требуется создавать, конфигурировать и рассылать в места использования только главный объект вместе с его иерархической структурой. Резервный объект формируется самой инфраструктурой ArchestrA (в частности, он рассылается отдельно от главного объекта). При резервировании каналов передачи данных нужно создавать, конфигурировать и пересылать в места использования как главный, так и резервный объекты DIObject (источники данных). Кроме того, нужно дополнительно создавать, конфигурировать и пересылать в места использования объект RedundantDIObject, который используется для управления переключением между двумя объектами-источниками данных.

В среде ArchestrA с резервированием активный и пассивный объекты непрерывно контролируют состояние друг друга, перехватывая управление в случае отказа парного объекта. В системе передачи данных контроль состояния объектов DIObject осуществляет объект RedundantDIObject, переключаясь между ними по мере необходимости.

Связи между объектами времени конфигурирования (главный/резервный) и времени исполнения (активный/пассивный) не фиксированы. Во время работы приложения в любой момент времени активным или пассивным может быть как главный, так и резервный объект. Если один из них переходит в активное состояние, второй автоматически становится пассивным.

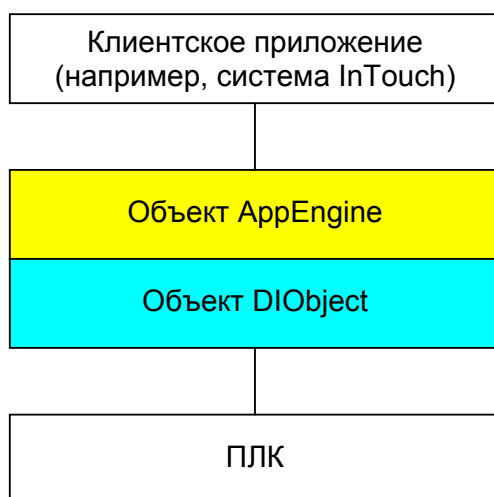
Параметры главного и резервного объектов могут быть изменены после их рассылки в места использования.

**Внимание!** При резервировании объекта AppEngine Archestra поддерживает топологию "один к одному", в которой компьютеры с главным и резервным объектами должны быть соединены кроссоверным кабелем и иметь фиксированные IP-адреса.

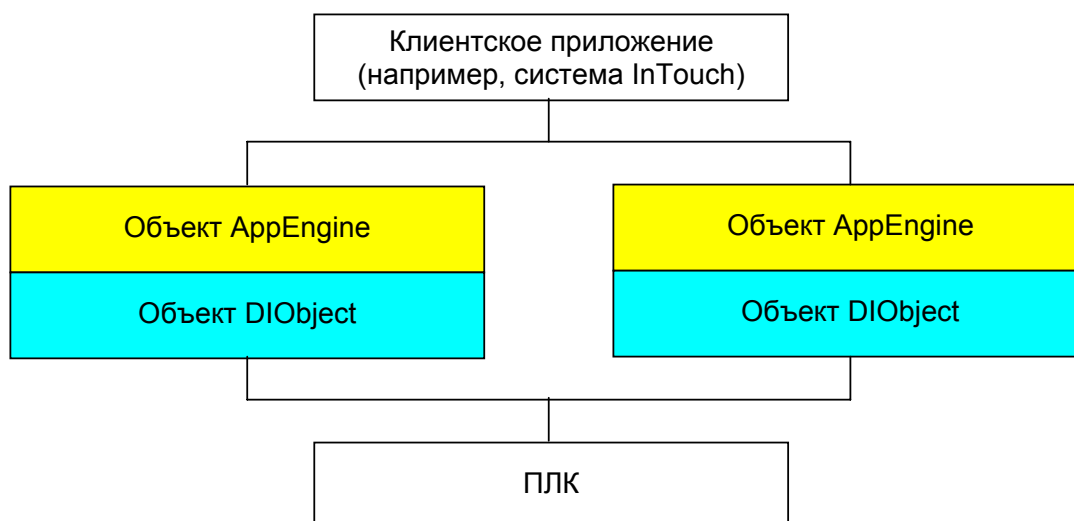
## Примеры конфигураций с резервированием

На следующих рисунках показаны системы без резервирования и с резервированием объектов AppEngine и объектов сбора данных.

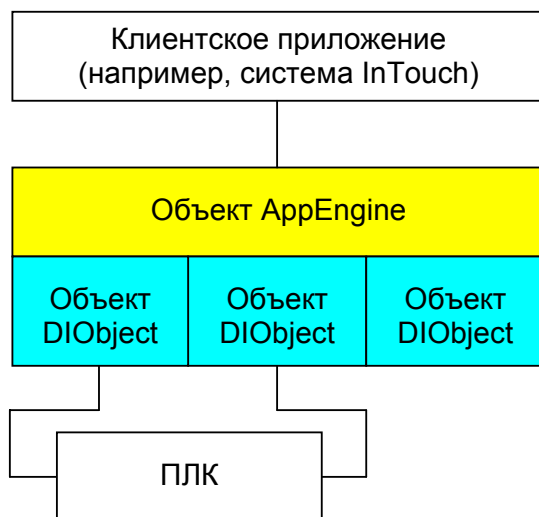
В следующей системе объекты AppEngine и объекты сбора данных не дублируются. Сбой компьютера с объектом AppEngine, отказ функционирования самого объекта AppEngine или объекта сбора данных приводит к разрыву соединения между клиентским приложением и ПЛК.



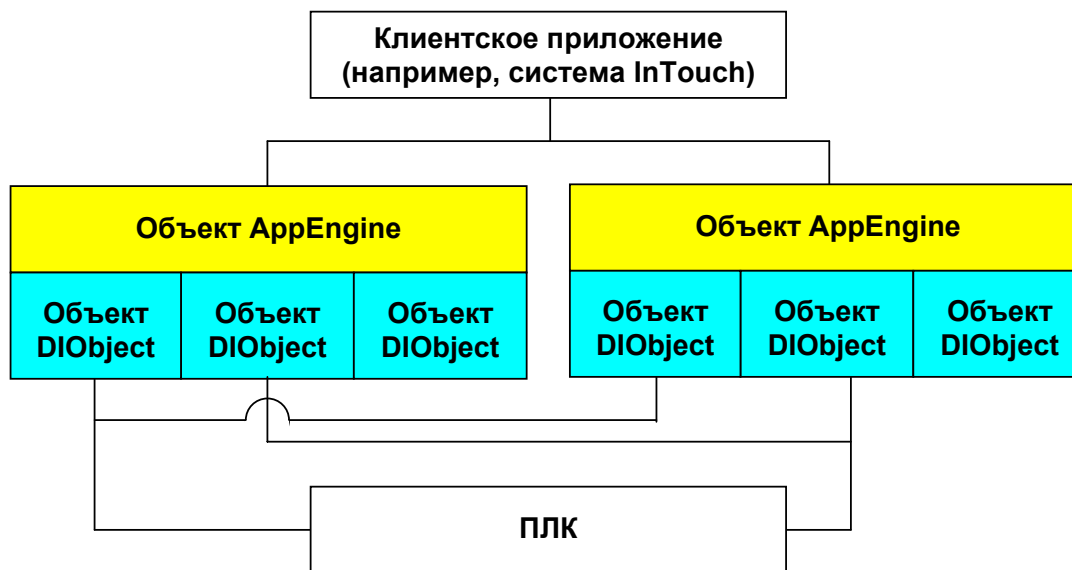
В следующей системе дублирован только объект AppEngine. Сбой компьютера с одним из объектов AppEngine или самого объекта не приводит к разрыву канала связи между клиентским приложением и ПЛК, поскольку в этом случае функционирование системы обеспечивает компьютер со вторым объектом AppEngine, который становится активным.



В следующей системе дублирован только объект передачи данных. При отказе функционирования объекта AppEngine взаимодействие клиентского приложения и ПЛК не прекратится, поскольку объект RedundantDIObject переключит объекты DIObject и обеспечит дальнейший обмен данными.



В следующей системе дублированы как объект AppEngine, так и объект DIObject. При отказе компьютера с одним из объектов AppEngine или самого объекта AppEngine взаимодействие клиентского приложения с ПЛК будет обеспечено вторым объектом AppEngine. При отказе активного объекта DIObject с активным объектом AppEngine взаимодействие клиентского приложения с ПЛК будет обеспечено резервным объектом DIObject.



## Резервирование объектов AppEngine

Резервирование объекта AppEngine задаётся в параметрах главного объекта. Нужно также сконфигурировать два объекта WinPlatform: один – для главного объекта AppEngine, другой – для резервного.

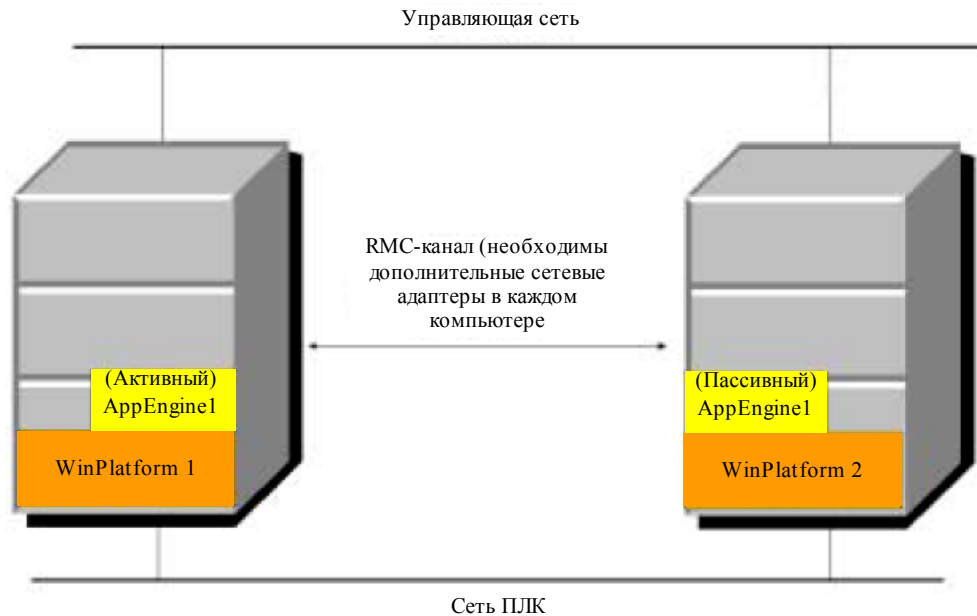
Конфигурации обоих объектов WinPlatform должны быть идентичными. Следующие параметры должны быть общими для обоих объектов:

- Каталоги промежуточного хранения при передаче данных.
- Пользовательские атрибуты.
- Скрипты.

**Примечание.** Во время конфигурирования объектов система позволяет связывать главный и резервный объекты AppEngine с одним и тем же объектом WinPlatform. Однако переслать их в место использования при

такой связи будет невозможно. Для этого их нужно связать с разными объектами WinPlatform.

На каждом компьютере производственной системы с дублированными объектами AppEngine должно быть установлено не менее двух сетевых адаптеров (см. рисунок ниже). Один из них должен обеспечивать подключение к управляющей сети (и сети ПЛК, если на компьютере установлено только два адаптера), ещё один – должен быть выделен для подключения соединяющего компьютеры кроссоверного кабеля Ethernet, представляющего собой резервный канал передачи сообщений (RMC – redundancy message channel). Он обеспечивает мониторинг резервирования, передачу сообщений и синхронизацию данных парных объектов AppEngine.



## Конфигурирование

Конфигурирование объектов AppEngine и WinPlatform выполняется в соответствующих редакторах объектов.

**Внимание!** Для обеспечения функций резервирования нужно, чтобы на компьютерах с объектами WinPlatform, в которых используются дублированные объекты AppEngine, были установлены одинаковые операционные системы.

Состояние каждого из объектов AppEngine после пересылки отражает как собственное состояние, так и состояние объекта-партнёра. Эти сведения отображаются в панели структуры приложения.

Возможны четыре типа состояния:

- **Использование пары (Pair Deployed):** используются как главный, так и резервный объекты.
- **Неиспользование пары (Pair Undeployed):** ни главный, ни резервный объекты не используются.
- **Частичное использование (Partial Deployed):** используется или главный, или резервный объект, тогда как объект-партнёр – не используется. Если объект AppEngine находится в состоянии частичного использования, тогда парный объект находится в состоянии частичного неиспользования.
- **Частичное неиспользование (Partial Undeployed):** не используется или главный, или резервный объект, тогда как объект-партнёр –

используется. Если объект AppEngine находится в состоянии частичного неиспользования, тогда парный объект находится в состоянии частичного использования.

---

**Внимание!** Подробнее о значках, отображающих состояние объектов, см. параграф "Создание объектов по шаблонам".

---

## Параметры резервирования объекта AppEngine

Параметры, определяющие резервирование объекта AppEngine, находятся на странице с закладками **Резервирование (Redundancy)** и **Общие (General)**.

В первую очередь нужно установить флажок **Разрешить резервирование (Enable Redundancy)** на странице **Резервирование (Redundancy)** свойств главного объекта AppEngine. Затем нужно соответствующим образом установить остальные параметры. Подробнее см. справочный файл данного типа объектов.

Введите в поле **Таймаут отказа (Engine Failure Timeout)** на странице **Общие (General)** значение 2000 мс. В дальнейшем, возможно, потребуется дополнительная настройка данного параметра (между 2000 мс и значением по умолчанию в 10000 мс) в зависимости от прикладных требований. Следует принимать во внимание, что состояние отказа фиксируется через период времени, в три раза превышающий указанную величину. При указании 2000 мс (2 с) аварийное переключение будет выполнено, если объект AppEngine не сможет взаимодействовать с приложением Bookstrap в течение 6 секунд. Значение по умолчанию (10000 мс или 30 с невозможности взаимодействия) может оказаться слишком большим для оптимального функционирования системы с резервированием.

После этого следует сбросить флажок **Перезапуск в случае отказа (Restart the Engine When it Fails)** на странице **Общие (General)**.

После того как конфигурация объекта будет сохранена, значок объекта изменится. В этот момент инфраструктура ArchestrA создаёт резервный объект AppEngine с той же конфигурацией, что и у главного объекта. Резервный объект находится в папке нераспределённых хостов (Unassigned Host) или в папке WinPlatform по умолчанию (если она указана в окне **Установка параметров пользователя – Configure User Information**).

При последующем изменении конфигурации главного объекта в фоновом режиме будет выполнен захват резервного объекта, изменение его конфигурации и возврат в систему без уведомления клиентских приложений.

Допускается создание шаблонов объектов AppEngine с указанием резервирования. Заблокировать флаг разрешения резервирования в шаблоне нельзя. При создании экземпляра объекта по этому шаблону будут созданы сначала главный, а затем резервный объекты.

После того как инфраструктура ArchestrA создаст резервный объект AppEngine, в главном объекте можно отключить дублирование. После сброса флажка резервирования и возврата объекта в систему, резервный объект будет удалён из Galaxy. Исключение составляет случай, когда резервный объект используется, – освобождение изменённого главного объекта и возврат его в систему завершится неудачей. Чтобы удалить из Galaxy резервный объект AppEngine, нужно предварительно отменить его использование.

---

**Внимание!** Определять резервирование объектов ApplicationObject, включённых в объект AppEngine, для которого эта функция указана, не обязательно. В момент системного сбоя объекты ApplicationObject и значения их атрибутов дублированы в пассивном объекте AppEngine,

---

который становится активным. Подробнее см. параграф "Действия, выполняемые системой в случае отказа одного из компонентов".

---

**Внимание!** Не устанавливайте флажок **Перезапуск в случае отказа (Restart the Engine When it Fails)** на странице **Общие (General)** окна свойств объекта при установленном флажке резервирования объекта AppEngine.

---

## Параметры резервирования объекта WinPlatform

В свойствах объекта WinPlatform устанавливаются следующие параметры, относящиеся к резервированию: **IP-адрес резервного канала передачи сообщений (Redundancy Message Channel IP Address)**, **Порт резервного канала передачи сообщений резервирования (Redundancy Message Channel Port)** и **Порт резервного главного канала (Redundancy Primary Channel Port)**. IP-адрес резервного канала передачи сообщений должен представлять собой фиксированный IP-адрес. Подробнее об этих параметрах см. в справочном файле для объектов WinPlatform.

---

**Внимание!** Чтобы пара объектов AppEngine могла успешно взаимодействовать друг с другом, нужно указать правильный порядок сетевых соединений в сетевых службах каждого компьютера. Рекомендуется каждому соединению давать понятные обозначения, такие как "Supervisory Net" или "Redundant Message Channel". Наименования соединений определяются в окне **Сетевые соединения (Network Connections)**. В этом же окне щёлкните **Дополнительные установки (Advanced Settings)** и с помощью кнопок с обозначениями стрелок задайте правильный порядок соединений. Первым в списке всегда должно идти соединение с управляющей сетью. Если на компьютере установлено несколько сетевых адаптеров (например для соединения с управляющей сетью, устройствами автоматике и для канала RMC), RMC-соединение может быть указано в любой позиции списка (за исключением первой).

---

**Внимание!** В параметрах адаптера управляющей сети нужно определить параметры DNS следующим образом: на странице **DNS** окна **Дополнительные параметры TCP/IP (Advanced TCP/IP Settings)** установить флажок **Зарегистрировать адреса этого соединения в службе DNS (Register this connection's addresses in DNS)**. В окне параметров адаптера RMC-соединения этот флажок нужно сбросить. Подробнее см. параграф "Компьютеры с несколькими сетевыми адаптерами".

---

## Структуры приложения



Главные и резервные объекты AppEngine в структурах приложения отображаются по-разному.

Главный объект отображается во всех структурах таким же образом, как и при отсутствии резервирования. Подробнее о структурах приложения см. в главах "Объекты" и "Редакторы объектов".

Резервный объект AppEngine отображается только в структуре использования и имеет то же имя, что и главный объект (с суффиксом "(Backup)"). Вместе с тем его значок отличается от значка главного объекта. Кроме того, никакие объекты ApplicationObject, входящие в состав главного объекта, не показываются. Это означает, что связать объекты ApplicationObject с резервным объектом AppEngine нельзя. Редактор резервного объекта AppEngine запускается в режиме "только просмотр". Если выделить резервный объект AppEngine в структуре использования и переключиться в другую структуру, соответствующий главный объект станет выделенным автоматически.

## Значки структуры приложения

В структуре приложения с резервированием объекты AppEngine отображаются следующими значками:

Значок	Объект
	Главный объект AppEngine
	Резервный объект AppEngine

Все остальные значки сохраняют то же своё значение, что и в системах без резервирования (см. главу "Введение").

## Команды ИСР

К главному объекту AppEngine применимы все команды, определённые для объектов ИСР. К резервному объекту применима только часть этих команд. Перечень допустимых команд приведён в следующей таблице.

Команда	Главный объект AppEngine	Резервный объект AppEngine
Assign To (Привязать)	+	+
Check In (Освободить)	+	-
Check Out (Захватить)	+	-
Delete (Удалить)	+	-
Deploy (Переслать)	+	+
Export (Экспортировать)	+	+
Galaxy Dump (Выгрузка Galaxy)	+	+
Galaxy Load (Загрузка Galaxy)	+	+
Go to Partner (Выделить парный объект)	+	+
Import (Импортировать)	+	+
Object Help (Справка по объектам)	+	+
Open (Открыть)	+	+
Open Read-Only (Открыть только для чтения)	+	+
Override Check Out (Игнорировать захват)	+	-
Properties (Свойства)	+	+



Rename (Переименовать)	+	–
Rename Contained Name (Изменить вложенное название)	+	–
Set As Default (Установить как умолчание)	+	–
Unassign (Отменить привязку)	+	+
Undeploy (Отменить использование объекта)	+	+
Undo Check Out (Отменить захват)	+	–
Upload Runtime Changes (Загрузка рабочих характеристик)	+	–
Validate (Проверить)	+	+
View in Object Viewer (Просмотр в Object Viewer)	+	+

Ниже приведены некоторые сведения о результатах выполнения указанных команд:

- В результате выполнения команды **Экспортирование (Export)** по отношению к главному или резервному объекту AppEngine будут экспортированы оба объекта.
- В результате выполнения команды **Импортирование (Import)** по отношению к главному или резервному объекту AppEngine будут импортированы оба объекта. При импортировании применяются стандартные правила разрешения конфликтов наименований.
- При выполнении команды **Дамп Galaxy (Galaxy Dump)** будут выведены сведения об обоих объектах.
- При выполнении команды **Загрузка Galaxy (Galaxy Load)** будет выполнена загрузка сведений об обоих объектах. При загрузке применяются стандартные правила разрешения конфликтов наименований.
- При выполнении команды **Загрузка рабочих характеристик (Upload Runtime Changes)** текущая конфигурация активного объекта AppEngine сначала будет записана в базу данных, а затем переписана в конфигурацию резервного объекта. Эта команда выполняется, когда используются или оба объекта, или только главный объект AppEngine.
- При выполнении команды **Переименование (Rename)** будут переименованы как главный, так и резервный объекты. Команда будет выполняться только тогда, когда и главный, и резервный объекты не используются. Переименовать только резервный объект AppEngine нельзя.
- Команда **Выделить парный объект (Go to Partner)** в меню правой кнопки позволяет быстро найти в структуре использования парный объект.

- При выполнении команды Просмотр в **Object Viewer (View in Object Viewer)** на экран выводятся сведения об активном объекте AppEngine независимо от того, какой объект (главный или резервный) был предварительно выделен.

## Ошибки и предупреждения

Выполнение некоторых требований (например порядка конфигурирования пары объектов в системе с резервированием) проверяется системной инфраструктурой. Если эти требования не выполняются, будут выдаваться сообщения об ошибках со следующей информацией:

- Несмотря на то, что необходимость резервирования может быть указана в параметрах объекта AppEngine до соответствующего конфигурирования объекта WinPlatform, будет выдано сообщение о том, что платформа (в частности, канал RMC) ещё не сконфигурирована.
- Если IP-адрес канала RMC для обеих платформ не будет указан, индикаторам состояния конфигурации главного и резервного объектов AppEngine будет присвоено значение "Error", а на экран будет выдано сообщение о том, что сетевой адаптер резервного соединения для WinPlatform не определён. После того как IP-адрес будет указан и объекты WinPlatform освобождены, система выполнит повторную проверку объектов AppEngine. Если объекты AppEngine при этом будут в состоянии захвата, проверка выполняться не будет.
- Если главный и резервный объект связаны с одним и тем же объектом WinPlatform, попытка переслать оба объекта AppEngine завершится ошибкой, а на экран будет выдано сообщение о том, что главный и резервный объекты AppEngine должны быть размещены на разных платформах. Необходимо указать для резервного объекта AppEngine другую WinPlatform и переслать его в место использования отдельно.
- Если сетевой адрес и IP-адрес канала RMC в параметрах объекта WinPlatform указывают на один и тот же адаптер, при сохранении данных будет выдано соответствующее предупреждающее сообщение. Эти параметры должны задавать различные сетевые адаптеры.

## Рассылка объектов AppEngine

Главный и резервный объекты AppEngine могут рассылаться как вместе, так и раздельно. При совместной рассылке (независимо от того, какой объект был в действительности указан для рассылки) главный объект всегда становится активным, а резервный – пассивным. При раздельной рассылке активным становится первый отосланный объект.

Встроенные объекты ApplicationObject всегда пересылаются в активный объект AppEngine. При пересылке первого объекта AppEngine из дублированной пары можно указать последовательную пересылку всех входящих в него объектов. Эта операция может выполняться одновременно с пересылкой главного и резервного объектов AppEngine.

---

**Внимание!** Если предполагается пересылка сначала резервного объекта AppEngine, затем входящих в него объектов, перед пересылкой главного объекта AppEngine следует удостовериться в работоспособности каналов связи с обоими рабочими компьютерами. В противном случае может возникнуть ошибка.

---

В рабочей системе активным или пассивным может быть как главный, так и резервный объект AppEngine (в зависимости от наличия условий сбоя).

Перед пересылкой главного и резервного объектов нужно проверить выполнение всех требований к их конфигурации. Каждый объект AppEngine должен быть связан с отдельной WinPlatform. Для каждой

платформы должен быть указан собственный резервный канал передачи сообщений RMS. Для совместной пересылки главного и резервного объектов в окне **Пересылка (Deploy)** нужно установить флажок **Включая резервный объект (Include Redundant Partner)**. Данный параметр будет недоступен при выполнении следующих операций:

- При каскадной пересылке из Galaxy.
- При пересылке нескольких выделенных объектов.
- При пересылке WinPlatform, содержащей главный или резервный объект AppEngine.

Сведения о выполнимости операций в зависимости от текущего состояния приведены в следующей таблице.

Состояние	Пересылка как главного, так и резервного объектов	Допустимость каскадной пересылки
WinPlatform, содержащая резервный объект AppEngine, сконфигурирована для аварийного переключения и используется	+	+
Резервный объект AppEngine в состоянии "Error"	+	+
WinPlatform, содержащая резервный объект AppEngine, не используется	–	+
WinPlatform, содержащая резервный объект AppEngine, не сконфигурирована для аварийного переключения и используется	+	+
Пересылка из узла Galaxy	–	+
Пересылка из WinPlatform с главным объектом AppEngine	–	+
Пересылка нескольких выделенных объектов	–	–
WinPlatform, содержащая резервный объект AppEngine, не сконфигурирована для аварийного переключения и не используется	–	+
Пересылка из резервного объекта AppEngine	+	+
Пересылка из главного	+	+

объекта AppEngine		
-------------------	--	--

Отмена использования дублированной пары объектов AppEngine аналогична отмене использования обычных объектов. Использование активного и резервного объектов AppEngine может быть отменено независимо друг от друга. Кроме того, можно отменить их использование как пары объектов, если выделить в структуре приложения один из них, выполнить команду **Отменить использование объекта (Undeploy)** и установить флажок **Включая резервный объект (Include Redundant Partner)**.

**Внимание!** Отмена использования объекта ApplicationObject, включая дублированные пары объектов AppEngine, не приводит к отмене установки программных модулей этих объектов на соответствующем компьютере. Установка программных модулей отменяется только при отмене использования WinPlatform.

## Действия, выполняемые системой в случае отказа одного из компонентов

При первоначальной пересылке дублированной пары объектов AppEngine сначала пересылаются программные модули и другие файлы главного объекта, затем файлы связанных с ним объектов ApplicationObject. После этого данные файлы по RMC-каналу пересылаются в пассивный объект AppEngine платформой активного объекта.

**Примечание.** Если какие-либо из этих файлов уже существуют в WinPlatform пассивного объекта AppEngine (возможно, связанные с другим объектом AppEngine в этой платформе), пересылаются только изменённые файлы.

Объекты ApplicationObject средой конфигурирования определяются всегда только для главного объекта AppEngine. Однако в среде исполнения они всегда сперва пересылаются в активный объект, независимо от того, является ли он главным или нет. Лишь после этого WinPlatform пересылает их из активного объекта в резервный объект AppEngine.

Во время исполнения активный и пассивный объекты AppEngine сначала пытаются установить связь по каналу RMC. Это происходит на начальном этапе функционирования одного из объектов в дублированной паре. Таким образом, если один из объектов пары будет позднее перемещён в другую платформу, канал связи между объектами может быть установлен заново.

Активный и пассивный объекты постоянно взаимодействуют друг с другом и контролируют состояние парного объекта.

При отказе аппаратных или программных средств компьютера активного объекта управление передаётся пассивному объекту, который становится активным. Чтобы удалить объект AppEngine, ставший пассивным, из компьютера, на котором он был размещён, нужно отменить использование этого объекта путём установки флажка **При отказе пометить как неиспользуемый (On failure mark as undeployed)** в окне **Отмена использования объекта (Undeploy)**, после чего связать его с другой WinPlatform, которая была определена на другом компьютере как резервная, и переслать его туда.

## Состояния дублированных объектов AppEngine

В любой момент времени дублированные пары объектов AppEngine могут находиться в одном из следующих состояний:

- **Активный (Active):** состояние объекта AppEngine, когда он установил связь с парным объектом, находящимся в одном из следующих

состояний: **Пассивный – не готов (Standby – Not Ready)**, **Пассивный – синхронизация с активным (Standby – Sync'ng with Active)** или **Пассивный – готов (Standby – Ready)**. Пассивный объект переходит в указанное состояние при обнаружении ситуации аварийного переключения. В данном состоянии объект AppEngine планирует и обеспечивает функционирование используемых объектов, пересылает данные контрольных точек и обновление списка подписчиков пассивному объекту AppEngine.

- **Активный – пассивный недоступен (Active – Standby not Available):** состояние объекта AppEngine, когда он определяет невозможность взаимодействия с парным объектом. Это значит: что изменения данных контрольных точек, списков подписчиков и состояний алармов не могут быть переданы пассивному объекту AppEngine; что регулярные тестовые пакеты от пассивного объекта не поступают; что получено уведомление о том, что пассивный объект прекращает или уже прекратил своё функционирование. Если объект AppEngine находится в этом состоянии, он 1) обеспечивает функционирование связанных с ним объектов ApplicationEngine в обычном режиме, 2) не может быть вручную переведён в пассивный режим и 3) продолжает попытки установления связи с пассивным объектом, но не передаёт ему никаких данных.
- **Определение состояния переключения (Determining Failover Status):** исходное состояние объекта AppEngine в начале функционирования. Он ещё не определил, является ли он активным или пассивным объектом AppEngine. Чтобы это определить, вначале предпринимаются попытки взаимодействия с парным объектом по каналу RMC, затем – по главной сети. Если связь в течение определённого периода времени не может быть установлена, объект AppEngine начинает функционировать как активный при условии, что он имеет все необходимые для этого программные модули и данные файла контрольной точки. В последующем этот объект AppEngine непрерывно пытается установить связь с парным объектом.
- **Пассивный – пропущены тестовые пакеты (Standby – Missed Heartbeats):** состояние объекта AppEngine, в которое он переходит, когда 1) в течение определённого периода времени не получает регулярные тестовые пакеты от активного объекта, 2) происходит отказ или "зависание" активного объекта и 3) выполняется плановое прекращение функционирования активного объекта AppEngine. В этом состоянии пассивный объект пытается определить, функционирует ли активный объект или нет. При ручном переключении объектов (с помощью атрибута ForceFailoverCmd) оно будет выполнено, только если регулярные тестовые пакеты не поступают из главной сети, но поступают по каналу RMC.
- **Пассивный – не готов (Standby – Not Ready):** состояние объекта AppEngine, в которое он переходит, когда 1) он теряет связь с парным объектом или сохраняет её, но пропускает изменения данных контрольной точки или изменения состояния аларма, передаваемые активным объектом, 2) в активный объект AppEngine были пересланы новые объекты, однако требуемые файлы на компьютере пассивного объекта AppEngine ещё не установлены, 3) связь пассивного объекта AppEngine с активным объектом по каналу RMC прервалась до завершения синхронизации данных. Как правило, парный объект AppEngine находится в одном из состояний **Активный – пассивный недоступен (Active – Standby not Available)**, **Активный (Active)** или **Пассивный – пропущены тестовые пакеты (Standby – Missed Heartbeats)**.
- **Пассивный – готов (Standby – Ready):** состояние объекта AppEngine после завершения синхронизации программных модулей и данных

контрольной точки с активным объектом AppEngine. В этом состоянии объект AppEngine проверяет, не произошёл ли отказ активного объекта, контролируя приход регулярных тестовых пакетов от активного объекта AppEngine, соответствуют ли требуемые для исполнения файлы файлам в активном объекте, и принимает следующую информацию от активного объекта: данные об изменении контрольной точки, уведомления, относящиеся подписке, сведения об изменениях в состояниях аларма, а также архивные блоки.

- **Пассивный – синхронизация с активным (Standby – Sync'ng with Active):** состояние объекта AppEngine, в котором он находится во время синхронизации своих данных с активным объектом. Если на компьютере пассивного объекта имеются программные модули, которые отсутствуют на компьютере активного объекта, они удаляются. И наоборот, если они есть на компьютере активного объекта, но отсутствуют на компьютере пассивного, они устанавливаются. После синхронизации состояния всех программных модулей объект AppEngine переходит в состояние **Пассивный – программные модули синхронизированы (Standby – Sync'd Code)**.
- **Пассивный – программные модули синхронизированы (Standby – Sync'd Code):** состояние пассивного объекта AppEngine, в котором он находится после успешной синхронизации всех программных модулей с активным объектом.
- **Пассивный – Данные синхронизированы (Standby – Sync'd Data):** состояние пассивного объекта AppEngine, в котором он находится после успешной синхронизации с активным объектом всех своих данных, включая данные контрольной точки и сведения для подписки. Как правило, объект из этого состояния переходит в состояние **Пассивный – готов (Standby – Ready)**.
- **Переключение в активное состояние (Switching to Active):** временное, переходное состояние пассивного объекта во время выполнения команды переключения в активное состояние.
- **Переключение в пассивное состояние (Switching to Standby):** временное, переходное состояние активного объекта во время выполнения команды переключения в пассивное состояние.
- **Неизвестное (Unknown):** состояние дублирующего парного объекта при потере связи между объектами AppEngine, или состояние объекта AppEngine, когда парный объект не функционирует.

---

**Внимание!** Перед перезагрузкой компьютера с одним из объектов AppEngine дублированной пары (как активным, так и резервным) нужно убедиться в наличии соединения с главной сетью. Если главная сеть будет отключена, при перезагрузке компьютера в качестве адреса сети будет использован IP-адрес канала RMC. Возникнет ситуация неверного дублирования с включением функций резервирования. Каждый раз при перезагрузке компьютера с резервированием нужно проверять правильность сетевых соединений. Подробнее см. параграф "Компьютеры с несколькими сетевыми адаптерами".

---

## Генерация алармов

При возникновении условий аварийного переключения ArcestrA передаёт сведения об алармах регистратору Logger. В их состав входит следующая информация.

- Имя объекта AppEngine, уведомившего о возникновении аларма.
- Имя узла, на котором находится этот объект.
- Состояние объекта AppEngine.

- Имя узла, на котором находится парный объект AppEngine.

**Внимание!** В зависимости от причины аварийного переключения, пассивный объект AppEngine может стать активным с отключенным сканированием, и поэтому генерация алармов может отсутствовать. При прекращении функционирования объекта AppEngine, когда сканирование отключено, блок контрольной точки может передать это состояние пассивному объекту, и позднее, когда этот объект станет активным, он начнёт функционирование в состоянии отключенного сканирования. Генерация алармов начнётся только после разрешения сканирования.

Система передаёт следующие сведения об алармах:

Аларм	Предшествующее состояние	Текущее состояние	Аларм генерируется в состоянии	Аларм сбрасывается в состоянии	Источник сведений об аларме
Standby Not Ready <sup>1)</sup> (Пассивный объект не готов)	Active	Standby – Not Ready	Standby – Not Ready	переход состояния Standby – Ready	Активный объект AppEngine
Standby Not Available (Пассивный объект недоступен)	Active	Active – Standby Not Available	Active – Standby Not Available	переход состояния Active	Активный объект AppEngine
Failover Occurred (Произошло аварийное переключение)			Пассивный объект становится активным	Во время следующего периода сканирования активного объекта AppEngine	Активный объект AppEngine

<sup>1)</sup> – Активный объект AppEngine контролирует состояние пассивного по каналу RMC, для того чтобы определить момент генерации этого аларма. Кроме того, если активный объект AppEngine находится в состоянии Активный – не готов (Standby – Not Ready), этот аларм не генерируется.

После аварийного переключения пассивный объект AppEngine, ставший активным, не будет передавать сведения об алармах, оставшихся от бывшего активного объекта. Состояние этих алармов в новом активном объекте AppEngine будет одним из следующих:

- Отсутствие аларма (Out of alarm).
- Не подтверждён (Unacknowledged).
- Не подтверждён – возврат в нормальное состояние (Unacknowledged – Return to normal).
- Подтверждён – возврат в нормальное состояние (Acknowledged – Return to normal).
- Подтверждён (Acknowledged).

Временные отметки также будут сохранены, включая следующие моменты:

- Подтверждения аларма.
- Состояние аларма изменилось на "true".
- Состояние аларма изменилось на "false".

Кроме того, сохраняется следующая информация:

- Сведения о подтверждении алармов.
- Сообщения, вводимые оператором при подтверждении алармов.

---

**Примечание.** Сведения обо всех алармах накапливаются и передаются пассивному объекту AppEngine в конце периода сканирования и до их отправки клиентским приложениям. Таким образом, пассивный объект при отказе активного может не получить сведений об алармах, возникших в промежутках между периодами сканирования. Порядок передачи информации об алармах гарантирует, что клиентские приложения не получают информацию об алармах, отличающуюся от той, которая была передана пассивным объектом AppEngine при отказе активного.

---

## Генерация архивной информации

Информация для занесения в архив генерируется всеми активными объектами (AppEngine и содержащимися в них) во время нормального функционирования в рабочем приложении.

Архивируемые сведения передаются архиватору только из активного объекта AppEngine.

Потеря связи с архиватором не приводит к аварийному переключению объектов. Активный объект AppEngine переходит в режим временного хранения передаваемых сведений, кэшируя данные каждые 30 секунд. Временно сохраняемая информация синхронизируется с пассивным объектом.

При аварийном переключении данные могут быть потеряны не более чем за 30-секундный интервал перехода объекта из пассивного состояния в активное. Это достигается сочетанием операций пересылки с промежуточным сохранением, когда архиватор недоступен, и обычными операциями передачи данных, когда архиватор доступен.

## Принудительное переключение

Система поддерживает принудительное переключение активного и пассивного объектов. Это выполняется с помощью атрибута ForceFailoverCmd объекта AppEngine. В частности, переключение можно инициировать с помощью скрипта или командой утилиты Object Viewer. Подробнее об использовании атрибута ForceFailoverCmd см. в справочном файле для объектов AppEngine.

## Резервирование каналов связи

Контроль дублированных источников данных DIObject осуществляется с помощью объекта RedundantDIObject. В отличие от объектов AppEngine, параметра резервирования у источников данных DIObject нет. Во всех состояниях они функционируют как автономные объекты.

В любой момент времени технологическую информацию через объект RedundantDIObject передаёт только один источник данных DIObject. При этом в обоих источниках должны быть одинаковым образом сконфигурированы параметры DIGroup, значения которого передаются при помощи объекта RedundantDIObject (который как раз и определяет, какой из двух источников данных является активным). Оба источника должны иметь одно и же адресное пространство элементов данных.

## Конфигурирование резервных источников данных

Конфигурирование источников данных с целью резервирования выполняется в ИСП с помощью соответствующего редактора объектов.



Состояния резервирования используемых объектов система сбора данных (два объекта DIObject и один RedundantDIObject) не определены.

## Конфигурирование объекта RedundantDIObject

Резервирование источников данных включает в себя настройку параметров всех трёх компонентов: главного и резервного источников данных DIObject и объекта RedundantDIObject.

Поскольку все эти объекты, по сути, представляют собой автономные объекты, операции, выполнимые с любыми другими объектами ApplicationObject, применимы и к ним. В их число входят все команды ИСР, команды выгрузки и загрузки Galaxy, а также операции импортирования и экспортирования.

Подробнее о настройке параметров этих объектов см. в соответствующих справочных файлах.

Основными действиями, которые выполняются с объектом RedundantDIObject, являются следующие:

- Установка параметров Главный источник данных (Primary DI Source) и Резервный источник данных (Backup DI Source) на странице Общие (General) редактора объектов.
- Определение общих групп сканирования и групп блочного чтения и записи на соответствующих страницах редактора объектов.

## Пересылка

Пересылка дублированных объектов накопления данных ничем не отличается от пересылки обычных объектов. Никаких особых условий для пересылки каждого из объектов DIObject и RedundantDIObject нет.

Подробнее о рассылке объектов в места использования см. главу "Объекты".

## Переключение источников данных в рабочем приложении

Все три объекта системы сбора данных с резервированием (DIObject и RedundantDIObject) с точки зрения использования, генерации алармов и архивирования данных функционируют, как и любой другой объект Arcestra. Они не имеют никаких связанных с поддержанием резервирования состояний и ограничений.

Во время работы приложения объект RedundantDIObject контролирует состояние источников данных DIObject и при необходимости осуществляет переключение с активного объекта на пассивный.



# Словарь терминов Arcestra

Ниже приведён перечень терминов, характерных для Arcestra и используемых во всём наборе документов.

## **приложение (application)**

Коллекция объектов в Репозитории Galaxy, выполняющих определённую задачу автоматизации. Синоним термина "Galaxy". В Репозитории Galaxy могут храниться несколько приложений.

## **объект ApplicationEngine (AppEngine)**

Модуль системы исполнения, содержащий в себе и исполняющий прикладную логику управления, определённую в объектах AutomationObject.

## **инструментальный набор ApplicationObject Toolkit**

Средства программирования, которые используются для разработки новых шаблонов ApplicationObject, включая их версии для времени конфигурирования и времени исполнения.

## **объект ApplicationObject**

AutomationObject, который представляет собой некоторый элемент приложения. Это может быть представление компонента автоматизированного процесса (например термопары, насоса, двигателя, вентиля, реактора или резервуара) или соответствующего прикладного компонента (например функционального блока, ПИД-контур, программных модулей на языках SFC или цепной логики, фазы серийного производства или спецификации статистического управления процессом).

## **зона (Area)**

Логическая группа AutomationObject, представляющих производственную зону или участок. Используется для логического выделения алармов, архивируемых данных и контроля доступа и представляется объектом Area.

## **назначение (assignment)**

Определение хоста для AutomationObject. Например для объекта AppEngine определяется объект WinPlatform.

## **атрибут (attribute)**

Элемент данных AutomationObject, доступный извне.

## **ссылка на атрибут (attribute reference string)**

Строка символов, однозначным образом указывающая на некоторый атрибут AutomationObject.

## **объект AutomationObject**

Программный объект, представляющий элемент производственного процесса (например аппаратные или

программные средства, модули исполнения) как именованный объект в Galaxy. Стандартное средство создания, именованя, загрузки, выполнения и контроля состояния соответствующего элемента.

**базовый шаблон (base template)**

Шаблон на верхнем уровне иерархии производных объектов. В отличие от других шаблонов этой иерархии, данный шаблон не является производным ни от какого другого шаблона, а разрабатывается в помощью инструментального набора ApplicationObject Toolkit и импортируется в Galaxy.

**группа блочного чтения (block read group)**

Блок DAGroup, который запускается пользователем или другим объектом. Он осуществляет считывание блока данных из внешнего источника с возвратом кода завершения.

**группа блочной записи (block write group)**

Блок DAGroup, который запускается пользователем или другим объектом после установки значений всех требуемых элементов данных. Блок данных пересылается во внешнее устройство. В конце блочной записи возвращается код завершения.

**журнал изменений (Change Log)**

Предыстория всех действий в ArchestrA, таких как создание объектов, освобождение и захват объектов, пересылка в места использования, сохранение, переименование, отмена использования, отмена захвата, игнорирование захвата и назначение.

**освобождение (Check In)**

Сохранение изменений конфигурации объекта в Репозитории Galaxy и предоставление доступа к нему другим пользователям.

**захват (check out)**

Монопольное овладение объектом для выполнения над ним каких-либо действий.

**контрольная точка (checkpoint)**

Набор данных о текущей конфигурации, состоянии и т.д., сохраняемых на диск для обеспечения автоматического перезапуска функционирующего AutomationObject. Перезапущенный объект будет обладать всеми характеристиками и значениями, которые он имел в момент сохранения контрольной точки на диск.

**вложенное название (contained name)**

Имя объекта внутри содержащего его контейнера. Например, объект "вентиль" с собственным названием "Valve101" может быть заключён внутрь объекта "реактор" и получить в контексте этого контейнера имя "Inlet". Вложенное имя должно быть уникальным среди всех имён объектов этого контейнера.

**вложение (containment)**

Концепция помещения одних AutomationObject внутри других, в результате чего создаётся иерархическая структура

объектов, отражающая прикладную модель и обеспечивающая большие возможности управления объектами. Помещённый внутри AutomationObject вложенный объект получает в дополнение к собственному имени новое (называемое иерархическим именем), которое состоит из имени контейнера и имени объекта в контексте контейнера. Например, объект "уровнемер" с собственным именем TIC101 может быть включен в состав объекта-контейнера с именем Reactor1 и получить вложенное имя "Level". Его Иерархическое имя будет выглядеть как "Reactor1.Level".

#### **блок DAGroup**

Блок доступа к данным, определяемый для объектов DeviceIntegration. Он определяет способ обмена информацией с внешними источниками данных. Может содержать сведения о подписке, а также о группах сканирования для блочного чтения и записи.

#### **сервер доступа к данным Data Access Server (DAServer)**

Программа-сервер, взаимодействующая по различным протоколам, включая протоколы OPC, DDE и SuiteLink, с объектами DINetwork и DIDevice в Archestra или со сторонними клиентскими приложениями.

#### **инструментальный набор Data Access Server Toolkit (DAS Toolkit)**

Инструментальный набор для разработки серверов DAServer.

#### **менеджер DAServer Manager**

Оснастка консоли управления MMC (Microsoft Management Console), которую предоставляет DAServer, и которая обеспечивает требуемый пользовательский интерфейс активизации, конфигурирования и диагностирования DAServer.

#### **пересылка в место использования (deployment)**

Операция создания AutomationObject на целевом компьютере. Включает в себя установку необходимых программных средств, сохранение на компьютере конфигурационных параметров объекта и запуск программного обеспечения объекта.

#### **создание производного объекта (derivation)**

Создание нового шаблона на основе существующего.

#### **производный шаблон (derived template)**

Любой шаблон, который был создан на основе другого существующего шаблона (родительского шаблона).

#### **объект DeviceIntegration (DIObject)**

AutomationObject, представляющий собой канал связи с внешним устройством. Объекты DIObject функционируют в составе объектов AppEngine и включают в себя объекты DINetwork и DIDevice.

#### **объект DIDevice**

Представление физического внешнего устройства (например ПЛК или терминального устройства), связанное с объектом DINetwork.

**объект DINetwork**

Представление физического соединения с объектом DIDevice через сервер доступа к данным DAServer.

**запись о событии (event record)**

Данные о системе, регистрируемые в журнале при возникновении определённого события (наподобие превышения аналоговой переменной максимального допустимого значения или подтверждения аларма).

**экспортирование (Export)**

Создание пакетного файла с расширением .aaPKG на основе информации, хранящейся в базе данных Galaxy. Полученный файл может быть импортирован в любую другую Galaxy с помощью реализованного в ИСП механизма импортирования.

**инфраструктура (framework)**

Инфраструктура ArchestrA, состоящая из общего набора служб, компонентов и интерфейсов, обеспечивающая создание и использование AutomationObject сбора, сохранения и визуализации производственных данных, управления, генерации отчётов, а также анализа накопленных данных.

**база данных Galaxy**

Реляционная база данных, в которой хранится конфигурационная информация обо всех объектах Galaxy.

**Репозиторий Galaxy**

Программная подсистема, состоящая из одной или нескольких баз данных Galaxy.

**Galaxy**

Приложение целиком. Комплектная ArchestrA, состоящая из одного пространства логических имён и коллекции объектов WinPlatform, AppEngine и AutomationObject. Один или несколько объединённых в сеть компьютеров, составляющих автоматизированную систему. Она определяет пространство имён, в котором существуют все компоненты и объекты, а также единый набор политик системного уровня, которым удовлетворяет функционирование всех компонентов и объектов.

**Иерархическое имя (hierarchical name)**

Полное имя вложенного объекта, включая собственное имя объекта-контейнера. Например, иерархическое имя объекта с вложенным именем "Inlet", содержащегося в контейнере с именем "Reactor1", будет выглядеть как "Reactor1.Inlet".  
Примечание. У объекта имеется также собственное имя, которое может отличаться от иерархического (например "Valve1").

**система архивирования (historical storage system)**

Система сохранения хронологических данных в сжатом виде, рассчитанная на большие объёмы информации. В качестве системы архивирования в IAS используется архиватор InSQL.

**хост (host)**

AutomationObject, с которым связываются другие объекты (например объект WinPlatform является хостом для объекта AppEngine).

**импортирование (import)**

Операция добавления в базу данных Galaxy шаблонов или экземпляров объектов из внешнего файла.

**сервер промышленных приложений IAS (Industrial Application Server)**

Название компонента в составе пакета FactorySuite, являющегося основой любого приложения. В его состав входят Репозиторий Galaxy, одна ИСР, платформа WinPlatform, объект AppEngine и базовая библиотека объектов ApplicationObject.

**экземпляр (instance)**

Сконфигурированное уникальным образом представление компонента приложения, созданное по шаблону (реализация шаблона)

**реализация (instantiation)**

Процесс создания нового объекта на основе соответствующего шаблона.

**ИСР (Integrated Development Environment)**

Набор разнообразных редакторов конфигурации, используемых для настройки всей системы, общих параметров инфраструктуры и разработки приложений.

**браузер журнала (Log Viewer)**

Оснастка консоли ММС, обеспечивающая возможность просмотра сообщений, передаваемых службе LogViewer.

**обмен сообщениями (Message Exchange)**

Система обмена сообщениями между объектами.

**объект (Object)**

Шаблон или реализация шаблона, существующая в БД Galaxy. Общая характеристика всех объектов заключается в том, что они хранятся в Репозитории Galaxy как отдельные компоненты.

**запрет сканирования (off scan)**

Состояние AutomationObject, обозначающее его бездействие и неготовность выполнять свои функции в приложении.

**разрешение сканирования (on scan)**

Состояние объекта AutomationObject, в котором он обычным образом выполняет функции в соответствии с установленным графиком.

**пакетный файл определений (Package Definition File (.aaPDF))**

Стандартный файл описаний, в котором записаны конфигурационные параметры и код реализации базового шаблона. Файл, как правило, имеет расширение .aaPDF.

**пакетный файл (Package File (.aaPKG))**

Файл, полученный в результате экспортирования данных из ИСР. В нём содержатся конфигурационные параметры и код

реализации шаблона и/или экземпляра. Файл, как правило, имеет расширение .aaPKG.

**ПЛК (PLC)**

Программируемый логический контроллер.

**свойства (properties)**

Характеристики, общие для всех атрибутов объекта, например имя, значение, качество и тип данных.

**ссылка (reference)**

Строка, обозначающая объект или данные его атрибута.

**группа сканирования (scan group)**

Блок DAGroup, для которого достаточно определить только интервал обновления, после чего данные будут извлекаться с указанной периодичностью.

**системная консоль управления (System Management Console (SMC))**

Средство централизованного администрирования исполнительной системой.

**собственное название (TagName)**

Уникальное имя экземпляра объекта, например "V1101".

**шаблон (template)**

Объект, содержащий набор конфигурационных параметров и программных модулей (необязательно), которые могут использоваться для создания новых объектов (включая производные шаблоны и экземпляры объектов).

**набор шаблонов (toolset)**

Именованная коллекция шаблонов, отображаемых вместе в панели шаблонов ИСР.

**объект UserDefined**

Объект AutomationObject, создаваемый по шаблону \$UserDefined. В этом шаблоне отсутствуют специфические для приложения атрибуты или программные модули (то есть пользователь может определить их самостоятельно).

**объект WinPlatform**

Объект, представляющий компьютер в Galaxy. В этот объект входят компонент общесистемного обмена сообщениями, набор базовых служб, операционная система и физические аппаратные средства. На базе объектов WinPlatform функционируют все объекты AppEngine.