ООО «БизнесИнтерСофт»

«Платформа ODANT. Команды API odaServer»

Руководство разработчика

2018

Оглавление

Общие сведения

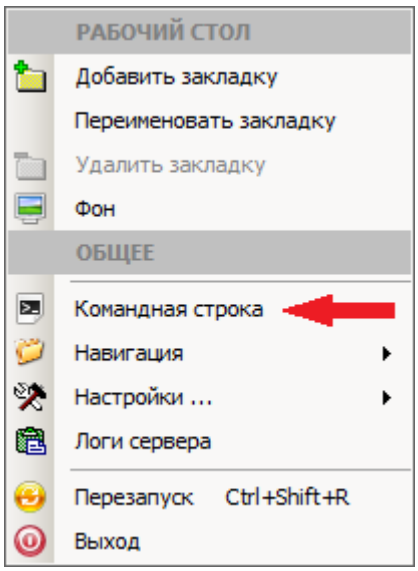
Данный документ является справочником по командам API сервера ODANT. После описания каждой команды приведен пример ее выполнения из командной строки в Windows-клиенте и из адресной строки Web-клиента. В приведенных примерах в качестве Web-клиента использовался браузер *Google Chrome*, часть команд выполнялась в программе *Postman*.

Команды API для Windows-клиента и Web-клиента имеют одинаковые имена и параметры, выполняют одни и те же действия, но могут возвращать отчет о выполнении команды в разных форматах. В этом случае в описании команды возвращаемое значение описывается два раза, отдельно для Windows-клиента и Web-клиента.

Выполнение команд в Windows-клиенте

По умолчанию командная строка в Windows-клиенте отключена, чтобы включить ее выполните следующие действия:

1) Правой клавишей мыши откройте контекстное меню рабочего стола Windows-клиента:



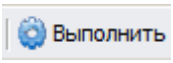
2) Выберите пункт Командная строка. Под рабочим столом появится форма для ввода командной строки:

Форма для ввода командной строки содержит следующие элементы:

Домен-организация: Test_1 – поле контекста команды. Это поле указывает элемент структуры базы данных, над которым выполняется команда. Чтобы задать контекст команды, перетащите мышкой в это поле соответствующий элемент структуры базы данных из проводника. Кнопка в правой части этого поля открывает список ранее использованных элементов структуры базы данных. Контекст для команды можно выбрать из этого списка.

 – текстовое поле для ввода команды и ее параметров. Кнопка в правой части этого поля открывает список ранее выполненных команд, что позволяет избежать повторного ввода этих команд вручную.

 – текстовое поле для ввода дополнительного параметра. Непосредственно в поле можно ввести только одну строку текста. Для ввода дополнительного параметра, содержащего большой объем текста, нажмите кнопку в правой части этого поля, чтобы открыть небольшой XML-редактор.



– кнопка запуска команды на выполнение.

 – кнопка закрытия командной строки.

Команда может иметь или не иметь параметров. Список параметров указывается после имени команды и отделяется от него символом «?» (вопросительный знак). Параметры в списке отделяются друг от друга символом «&» (амперсанд). Для каждого параметра указывается его имя и значение. Имя параметра и значение соединяются символом «=» (знак равенства). Поскольку для каждого параметра указывается его имя, порядок параметров в команде не имеет значения.

Примечание – Символы пробела между элементами команды не допускаются, т.к. они рассматриваются как часть имени команды, имени параметра или значения параметра. Символы пробела можно использовать только как часть значения параметра.

В общем случае команда в командной строке должна иметь следующий формат:

```
commandName?parameter1=value1&parameter2=value2&…&parameterN=valueN
```

где: `commandName` – имя команды,
`parameter` – имя параметра,
`value` – значение параметра.

Кроме параметров, указанных непосредственно после имени команды, в команду передаются параметры из поля контекста команды и из поля дополнительного параметра.

Поле контекста команды можно не использовать, а передавать контекст в параметре с именем ***id***, значение которого имеет приоритет над значением, содержащемся в упомянутом поле. Передача контекста через поле более наглядно, т.к. в поле указывается наименование элемента структуры базы данных в том же виде, как и в проводнике, а в параметре ***id*** указывается полный цифровой идентификатор этого элемента. Параметр ***id*** позволяет адресовать более мелкие элементы структуры базы (например, отдельные объекты, отдельные файлы и т.д.). В примерах, приведенных для команд, там, где это возможно, контекст передается через поле контекста, поскольку это обеспечивает более наглядную ссылку на элемент структуры базы данных, над которым выполняется команда.

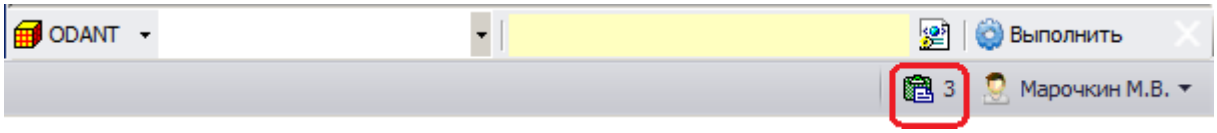
Примечание – В параметре id в префиксах идентификаторов и в самих 16-ричных идентификаторах разрешается использовать только прописные буквы латинского алфавита.

В результате выполнения команды, как правило, вносятся изменения в структуру базы данных или в состояние элементов базы данных, либо возвращается информация, о каком-либо элементе базы данных. Кроме того, краткий отчет о выполнении команды отображается в окне консоли. В примерах, приведенных для команд, приводятся

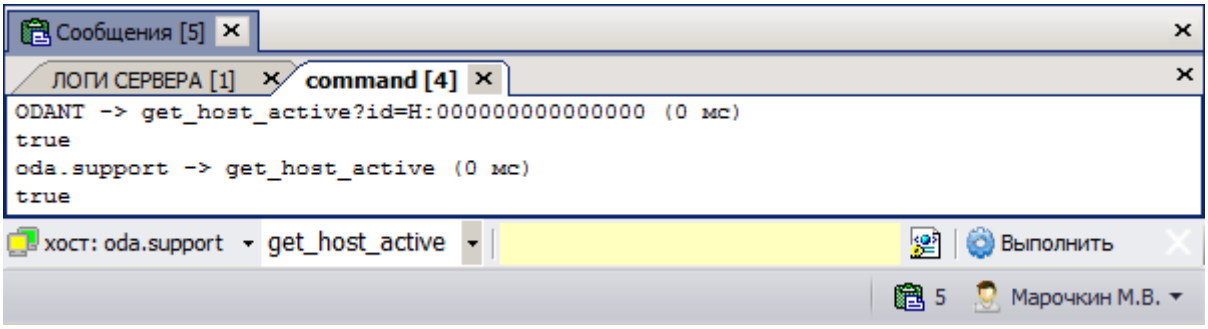
пояснения, где и как можно посмотреть результат выполнения команды. Для всех команд приводятся сообщения, выдаваемые в окно консоли.

Окно консоли содержит сообщения, выдаваемые различными компонентами платформы ODANT, в том числе и сообщения командной строки. Сообщения командной строки находятся на закладке **command** окна консоли.

Чтобы открыть окно консоли щелкните мышкой по индикатору сообщении расположенному на панели состояния под командной строкой, справа от имени текущего пользователя. Индикатор сообщений отображается только при наличии сообщений от компонент платформы ODANT.



Окно консоли располагается над командной строкой и имеет вид:



Результат выполнения команды отображается в окне в виде двух и более строк. Первая строка всегда содержит текст выполненной команды. Вторая и последующие строки содержат результат выполнения команды. Текст выполненной команды формируется из значений, находившихся в полях командной строки на момент выполнения команды, и имеет формат:

```
context -> command_and_parameters (XXX мс)
```

где:

`context` – наименование элемента структуры базы данных, на который ссылалось поле контекста команды;

`command_and_parameters` – команда с параметрами в том виде как она была задана в поле для ввода команды;

`XXX` – время в миллисекундах, которое потребовалось на выполнение команды.

Выполнение команд в Web-клиенте

В Web-клиенте команда с параметрами записывается конце адресной строки. Адресная строка с командой имеет один из следующих форматов:

```
URL_сервера/FullID?command_with_parameters
        или
URL_сервера?command_with_parameters
```

где `URL_сервера` – URL-адрес сервера, на котором расположен ODANT-хост;

`FullID` – полный идентификатор структурного элемента, являющегося контекстом для команды. `FullID` может отсутствовать в адресной строке. В этом случае среди параметров команды должен находиться параметр ***id***, содержащий полный идентификатор структурного элемента.

`command_with_parameters` – команда с параметрами.

`URL_сервера` и `FullID` вместе образую Web-адрес ODANT-элемента.

Команда с параметрами пишется в конце адресной строки и в общем случае имеет следующий формат:

```
?method=commandName&parameter1=value1&parameter2=value2&…&parameterN=valueN
```

где: `method` – обязательный параметр для ввода имени команды
`commandName` – имя команды,
`parameter` – имя параметра,
`value` – значение параметра.

Команда может иметь или не иметь параметров. Команда отделяется от Web-адреса ODANT-элемента символом «**?**» (вопросительный знак). Параметры в списке отделяются друг от друга символом «**&**» (амперсанд). Для каждого параметра указывается его имя и значение. Имя параметра и значение соединяются символом «**=**» (знак равенства). Поскольку для каждого параметра указывается его имя, порядок параметров в команде не имеет значения.

Примечания

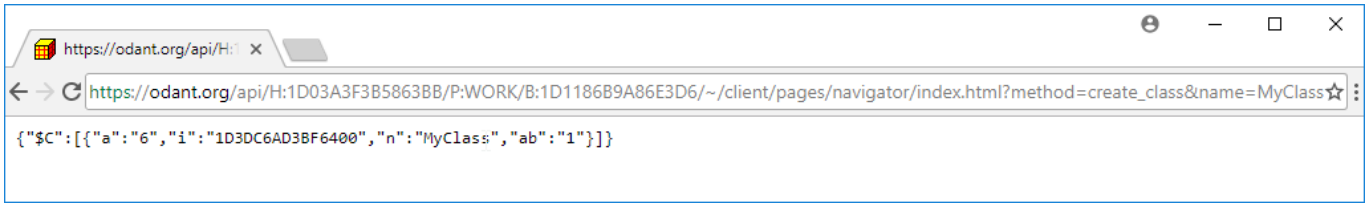
1) Символы пробела между элементами команды не допускаются, т.к. они рассматриваются как часть имени команды, имени параметра или значения параметра. Символы пробела можно использовать только как часть значения параметра.

2) С точки зрения синтаксиса адресной строки, имя команды является значением параметра *method*. Поэтому параметр *method* с именем команды можно указывать в любом месте команды, в том числе после параметров команды.

Команда использует контекст, передаваемый в параметре ***id***. Если параметр ***id*** не задан, то команда выделяет контекст из Web-адреса ODANT-элемента. Использование контекста из Web-адреса ускоряет ввод команды, кроме того, во многих случаях можно сразу наблюдать результат выполнения команды, поскольку в окне браузера отображается тот элемент, над которым или в котором команда производит действие. В примерах, приведенных для команд, там, где это возможно, контекст берется из Web-адреса, поскольку это делает текст примера более коротким и наглядным.

Примечание – В параметре ***id*** в префиксах идентификаторов и в самих 16-ричных идентификаторах разрешается использовать только прописные буквы латинского алфавита.

В результате выполнения команды, как правило, вносятся изменения в структуру базы данных или в состояние элементов базы данных, либо возвращается информация, о каком-либо элементе базы данных. Кроме того, в браузере открывается страница с кратким отчетом о выполнении команды. В примерах, приведенных для команд, приводятся пояснения, где и как можно посмотреть результат выполнения команды. Для всех команд приводятся сообщения, выдаваемые в окно браузера. Если сервер не распознал команду, то страница с сообщениями не открывается, никакие сообщения не выдаются, и текущая страница в окне браузера не изменяет свой вид. Пример команды и отчета о ее выполнении:



Автоматически формируемая адресная строка, помимо полного идентификатора элемента, содержит ссылку на визуальное представление этого элемента. Ссылка на визуальное представление следует после полного идентификатора и отделена от него символами “***/~/***” (прямой слеш, тильда, прямой слеш). На рисунке выше ссылка на визуальное представление домена-базы имеет вид:

```
/~/client/pages/navigator/index.html
```

Если команда не имеет параметра ***id***, т.е. выделяет идентификатор элемента из Web-адреса, то команда может воспринимать ссылку на визуальное представление как часть идентификатора элемента. Поэтому в примерах, приведенных для команд, мы удаляли эту ссылку из адресной строки перед вводом команды. Кроме того удаление ссылки делает текст примера более коротким и наглядным.

Автоматически формируемая адресная строка всегда содержит полный идентификатор структурного элемента уровня раздела, домена или класса, отображаемого в окне браузера. Этот идентификатор может использоваться как контекст команды. Идентификаторы для более мелких структурных элементов автоматически не формируется, т.к. они не имеют собственного визуального представления для отображения в браузере, поэтому для их адресации мы использовали в команде параметр ***id***.

Примечание – Вместо использования параметра ***id*** для адресации мелких структурных элементов можно дописать недостающую часть идентификатора в Web-адресе элемента, но это усложнит описание примера.

При составлении примеров команд мы следовали следующему правилу:

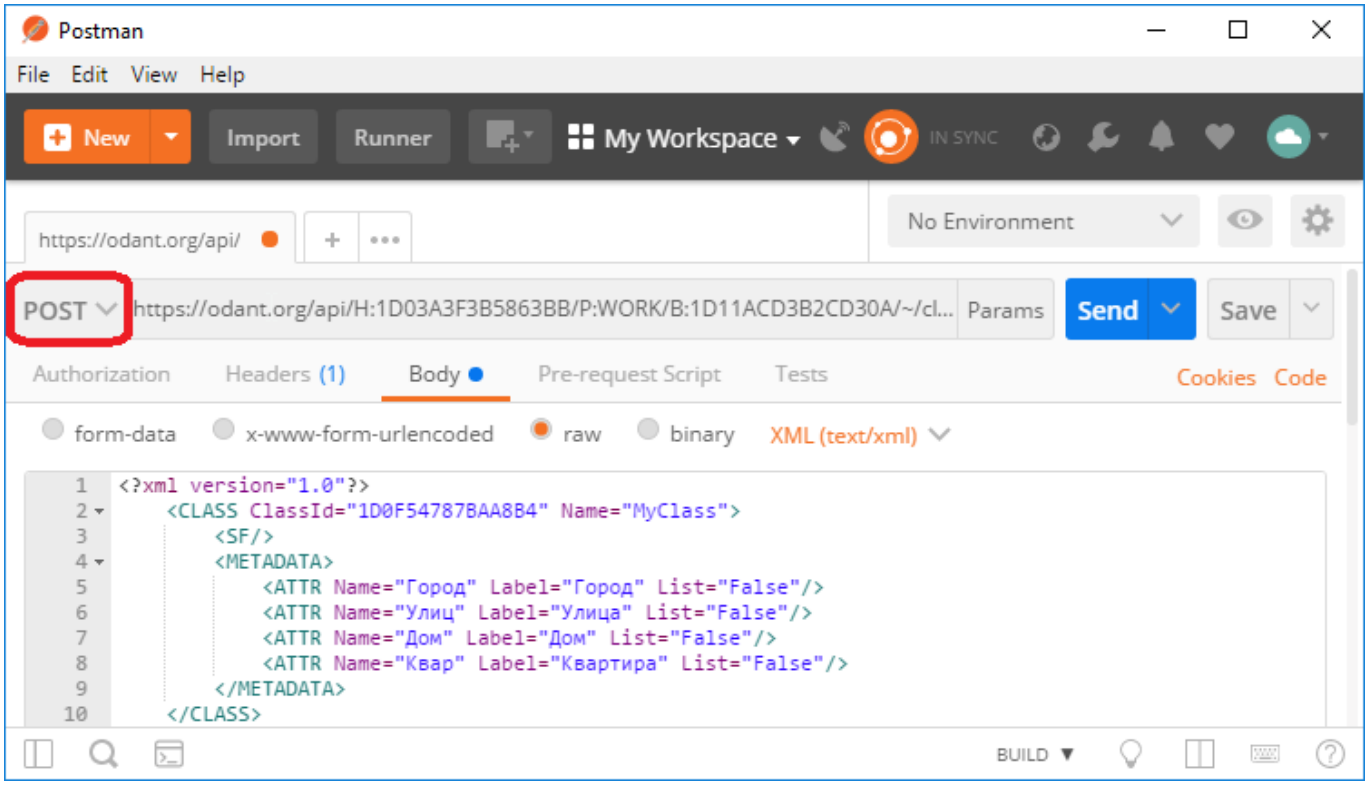
- открывали в окне браузера требуемый структурный элемент базы данных;
- удаляли ссылку на визуальное представление элемента из адресной строки;
- дописывали команду и ее параметры в конец адресной строки;
- использовали в команде параметр ***id***, только если идентификатор в Web-адресе не соответствовал требуемому структурному элементу.

Выполнение команд в программе Postman

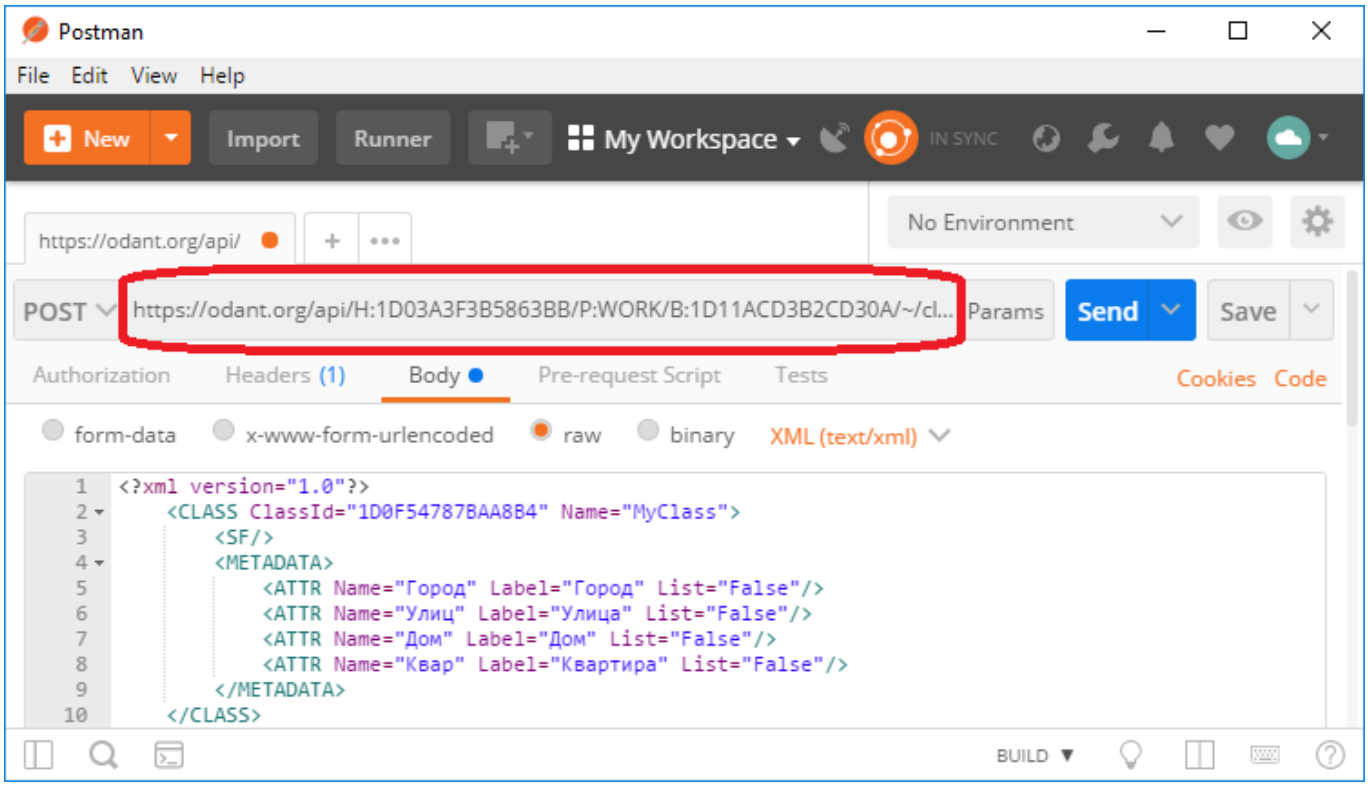
В некоторые команды требуется передавать большой объема текстовой информации (например, список идентификаторов, XML-документ, текст XQuery-запроса). Естественно передавать эту информацию через адресную строку браузера невозможно. Поэтому для выполнения таких команд используется программа Postman.

Ввод команды в программе Postman осуществляется следующим образом:

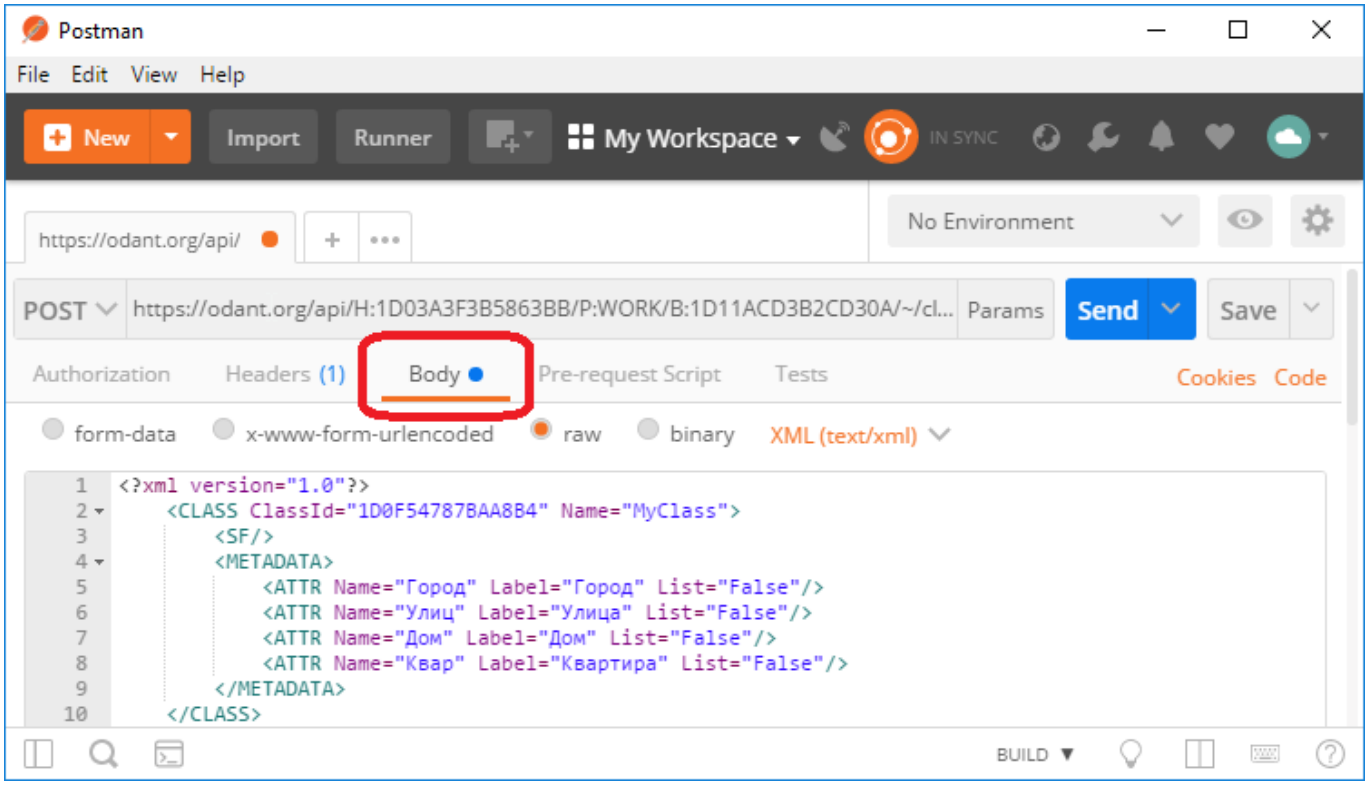
1) Установите тип запроса *POST*:



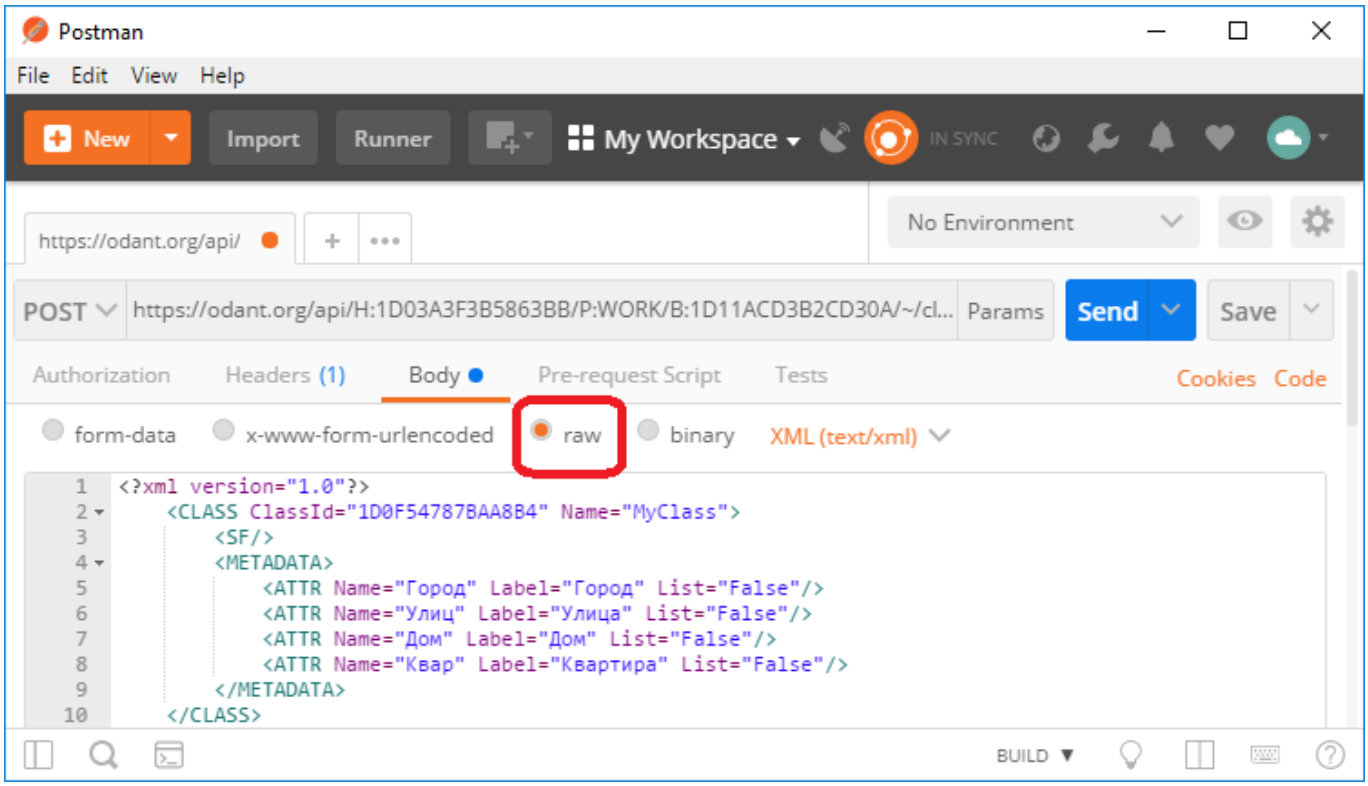
2) Введите адрес и команду в адресную строку (формат команды описан выше):



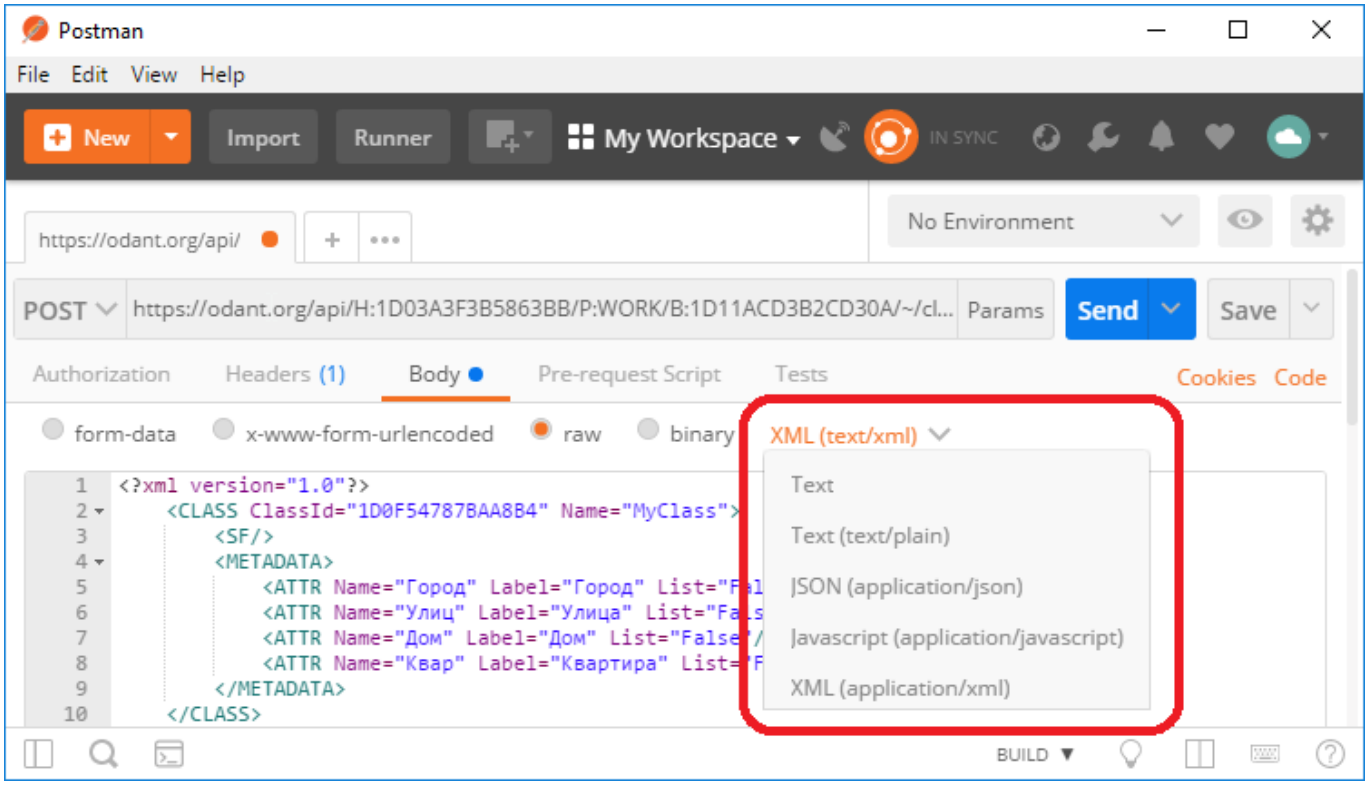
3) Для ввода дополнительной информации для команды, откройте текстовый редактор. Редактор открывается кнопкой *Body*, расположенной под адресной строкой:



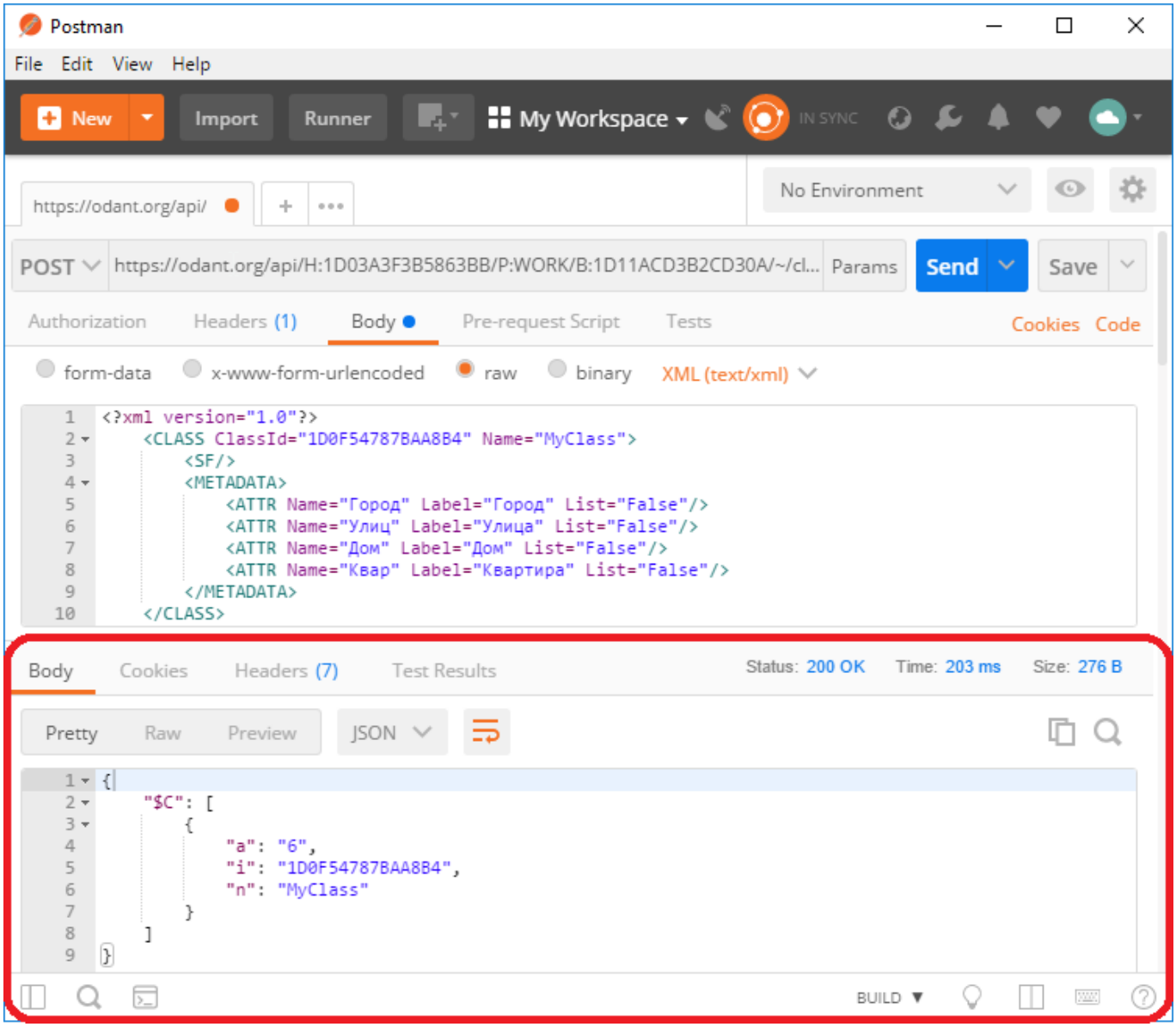
затем нажмите кнопку *raw*:



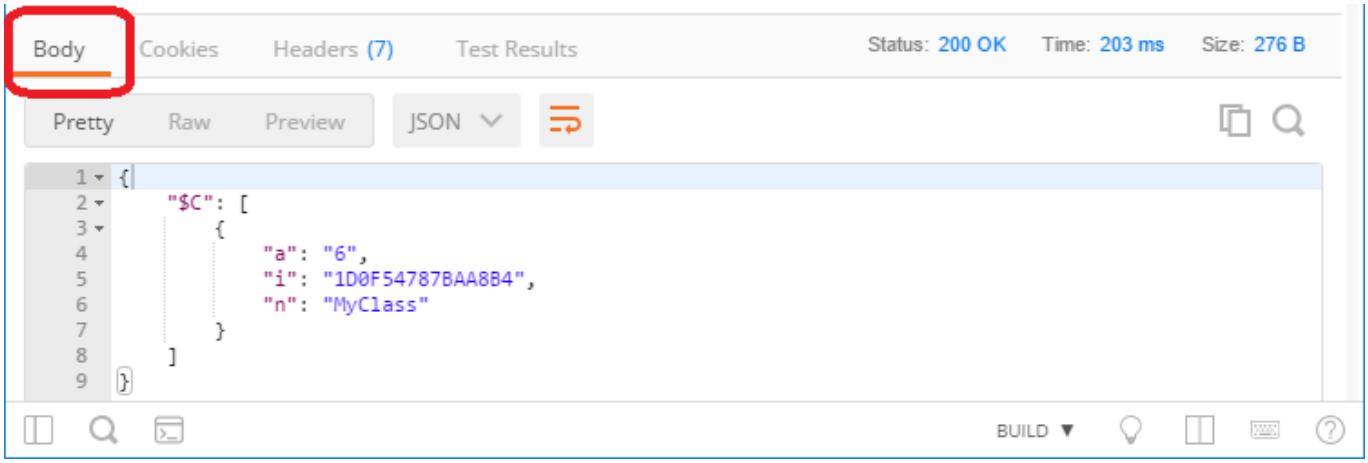
в редакторе можно выбрать требуемый режим подсветки синтаксиса из выпадающего списка:



4) Нажмите кнопку *Send*, чтобы послать команду на выполнение. Отчет о выполнении команды отобразится в окне консоли, расположенном под редактором:

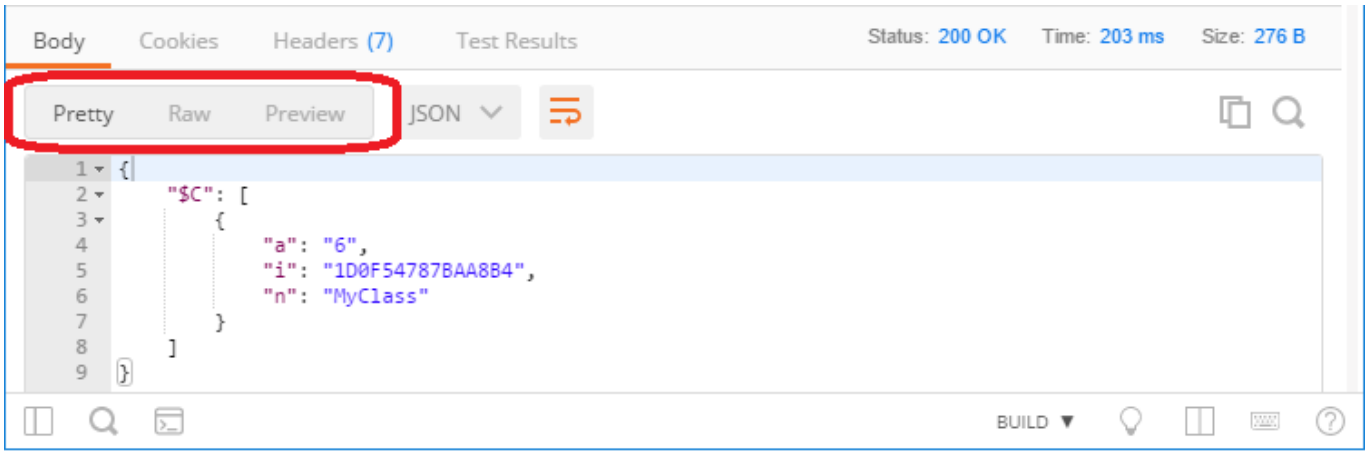


В окне консоли могут отображаться несколько типов сообщений, возвращаемых сервером. Чтобы отображался результат выполнения команды, выберите режим *Body*, нажав на одноименную кнопку, расположенную в заголовке окна консоли:

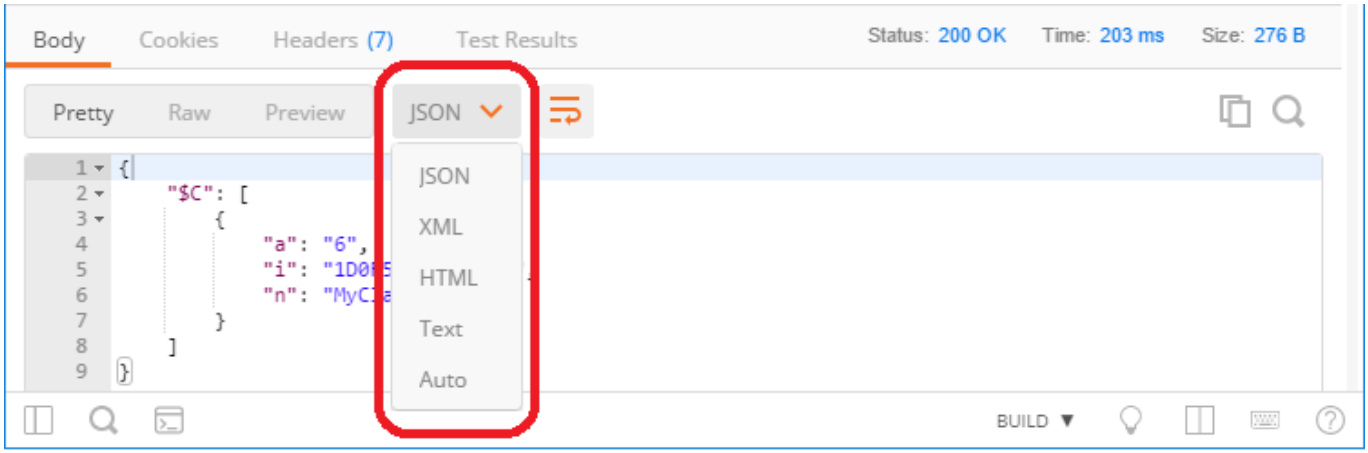


С целью улучшения зрительного восприятия сообщения, в окне консоли можно переключать визуальное представление сообщений с помощью кнопок *Pretty*, *Raw* и

Preview. В режиме *Raw* значение, возвращаемое командой, представлено «как есть», и совпадает с тем, что мы видим в окне Web-браузера (обычно это одна очень длинная строка). В режиме *Pretty* возвращаемое значение подвергается визуальному форматированию, которое выделяет структурные элементы сообщения с помощью подсветки синтаксиса, добавления разделяющих пробелов и взаимного расположения элементов на экране:

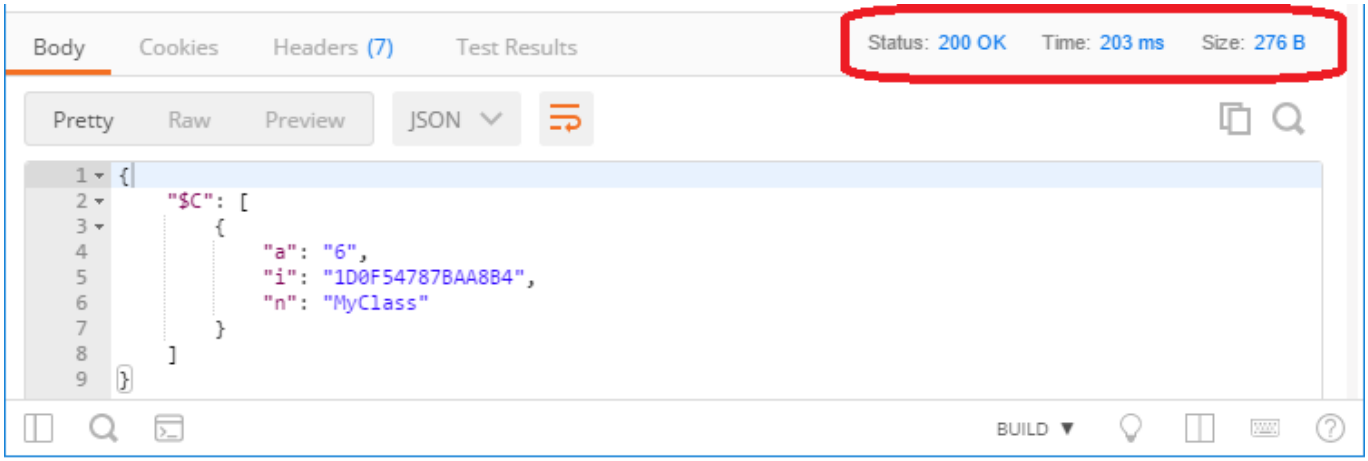


В режиме *Pretty* из выпадающего списка можно выбрать режим подсветки синтаксиса:



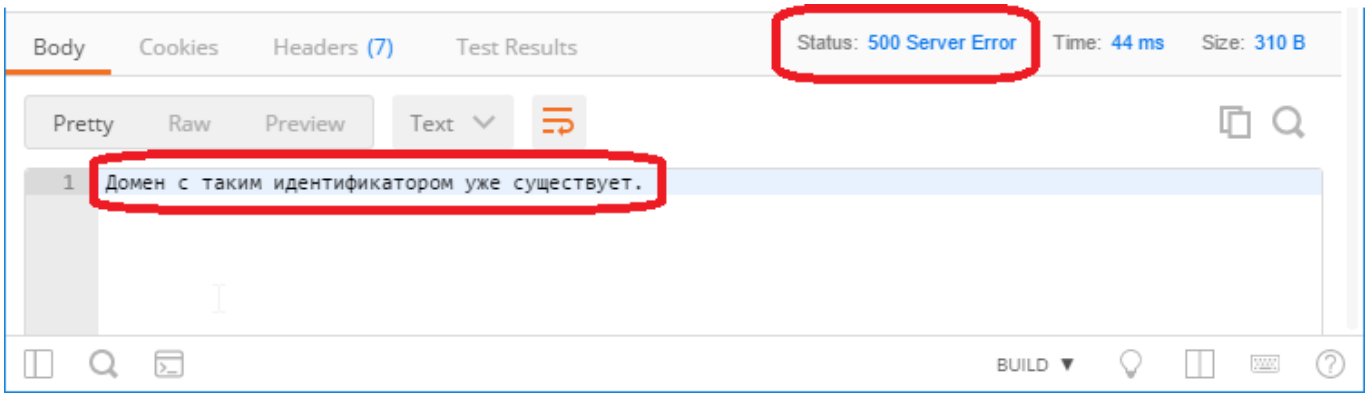
В примерах команд, выполненных в программе Postman, для просмотра значений, возвращаемых командами, используется режим *Pretty*, как наиболее наглядный.

Также в окне консоли отображается информация о реакции сервера на запрос:



В поле *Status* отображается цифровой код ответа сервера и его текстовая мнемоника (в примере код ответа «*200*», мнемоника кода «*OK*», т.е. команда выполнена успешно). В поле *Time* отображается время выполнения запроса (в примере 203 мс). В поле *Size* отображается размер ответа от сервера, включая служебный заголовок (в примере 276 байт).

Если сервер не смог выполнить запрос, то в поле *Status* отображается код ответа отличный от «*200*», а в окне консоли отображается текст сообщения об ошибке.



Примечание – Если команда изменяет структуру базы данных, то посмотреть эти изменения можно только в Web-браузере или Windows-клиенте.

Возвращаемое значение

Практически все команды API сервера возвращают какое-то значение. Команды API сервера, запускаемые в Widows-клиенте и Web-клиенте, выполняющие одни и те же функции, возвращают одни и те же значения, но в разном формате.

Возвращаемое значение в Windows-клиенте

В Windows-клиенте формат возвращаемого значения определяется самим значением. Например, если возвращается логическое значение, то выдается строка «true» или «false», если возвращается список идентификаторов, то выдается строка из 15-разрядных чисел разделенных пробелами или другими разделителями.

При возникновении ошибки, команда в Windows-клиенте возвращает строку с описанием возникшей ошибки. Строка начинается с заголовка «***~Error~***» (но не всегда), за которым следует описание ошибки. Например:

```
MyBase [Мой компьютер] -> create_domain (1 мс)
~Error~Домен с таким идентификатором уже существует.


MyDomain [Мой компьютер/MyBase] -> delete_domain (0 мс)
error context
```

Возвращаемое значение в Web-клиенте

В Web-клиенте возвращаемое значение всегда является структурой в JSON- или XML-формате, причем можно выбрать формат, в котором будет возвращаться значение. Для выбора формата необходимо в команде использовать параметр ***format***. Параметр ***format*** может принимать следующие значения: xml, xml-standart, json-1, json-2, json-3,

json-4, json-5, json-6, json-7. Если указать в параметре ***format*** значение json или вообще опустить этот параметр, то будет использоваться формат по умолчанию json-7.

Примечание – Чтобы сократить объем документа:
1) в описании команд параметр ***format*** не указывается;
2) описание возвращаемого командой значения дается только для JSON-формата;
3) все команды в примерах возвращают значение только в JSON-формате.

Рассмотрим множество форматов на базе JSON более подробно. Наличие нескольких форматов связано с тем, что потребовалось адаптировать формат JSON под передачу XML-документов, которые имеют более сложную структуру, чем допускает стандарт JSON. Поэтому в JSON-сообщение добавляется информация о том, чем являются передаваемые значения для XML-документа: атрибутами или узлами. Значение параметра ***format*** указывает, каким способом включить эту дополнительную информацию в JSON-сообщение.

Рассмотрим, как отображается XML-документ в разные форматы JSON-сообщений.

Исходный XML-документ

```
<A B='attr_B' b='attr_b'>
     <B C='attr_c'>
          <C a='111' b='222' c='333' d='444'/>
          <C a='aaa' b='bbb' c='ccc' d='ddd'/>
     </B>
     <B x='ccc' C='ffg' y='vvv' z='bbb'>
          <C q='qqq' w='www' c='ccc' r='rrr'/>
     </B>
</A>
```

Формат json-1

Перед именем атрибута всегда ставится префикс **$** (символ доллара).

```
{
   "A": [{
      "$B": "attr_B",
      "$b": "attr_b",
      "B": [{
           "$C": "attr_c",
           "C": [{
                "$a": "111",
                "$b": "222",
                "$c": "333",
                "$d": "444"
           }, {
                "$a": "aaa",
                "$b": "bbb",
                "$c": "ccc",
                "$d": "ddd"
           }]
      }, {
           "$x": "ccc",
           "$C": "ffg",
           "$y": "vvv",
```

```
            "$z": "bbb",
            "C": [{
                 "$q": "qqq",
                 "$w": "www",
                 "$c": "ccc",
                 "$r": "rrr"
            }]
      }]
    }]
}
```

Формат json-2

Перед именем атрибута ставится префикс **$** (символ доллара), только если существует узел с аналогичным именем.

```
{
    "A": [{
      "$B": "attr_B",
      "b": "attr_b",
      "B": [{
            "$C": "attr_c",
            "C": [{
                 "a": "111",
                 "b": "222",
                 "c": "333",
                 "d": "444"
            }, {
                 "a": "aaa",
                 "b": "bbb",
                 "c": "ccc",
                 "d": "ddd"
            }]
      }, {
            "x": "ccc",
            "$C": "ffg",
            "y": "vvv",
            "z": "bbb",
            "C": [{
                 "q": "qqq",
                 "w": "www",
                 "c": "ccc",
                 "r": "rrr"
            }]
      }]
    }]
}
```

Формат json-3

Атрибуты группируются в ключе с именем **$** (символ доллара).

```
{
    "A": [{
      "$": {
           "B": "attr_B",
           "b": "attr_b"
      },
      "B": [{
           "$": {
                "C": "attr_c"
           },
           "C": [{
                "$": {
                     "a": "111",
                     "b": "222",
                     "c": "333",
                     "d": "444"
                }
           }, {
                "$": {
                     "a": "aaa",
                     "b": "bbb",
                     "c": "ccc",
                     "d": "ddd"
                }
           }]
      }, {
           "$": {
                "x": "ccc",
                "C": "ffg",
                "y": "vvv",
                "z": "bbb"
           },
           "C": [{
                "$": {
                     "q": "qqq",
                     "w": "www",
                     "c": "ccc",
                     "r": "rrr"
                }
           }]
      }]
    }]
}
```

Формат json-4

Узлы группируются в ключе с именем **$$** (два символа доллара).

```
{
    "$$": {
      "A": [{
           "B": "attr_B",
           "b": "attr_b",
```

```
            "$$": {
                 "B": [{
                      "C": "attr_c",
                      "$$": {
                           "C": [{
                                "a": "111",
                                "b": "222",
                                "c": "333",
                                "d": "444"
                           }, {
                                "a": "aaa",
                                "b": "bbb",
                                "c": "ccc",
                                "d": "ddd"
                           }]
                      }
                 }, {
                      "x": "ccc",
                      "C": "ffg",
                      "y": "vvv",
                      "z": "bbb",
                      "$$": {
                           "C": [{
                                "q": "qqq",
                                "w": "www",
                                "c": "ccc",
                                "r": "rrr"
                           }]
                      }
                 }]
            }
       }]
    }
}
```

Формат json-5

Атрибуты группируются в ключе с именем **$** (символ доллара), узлы группируются в ключе с именем **$$** (два символа доллара).

```
{
    "$$": {
      "A": [{
            "$": {
                 "B": "attr_B",
                 "b": "attr_b"
            },
            "$$": {
                 "B": [{
                      "$": {
                           "C": "attr_c"
                      },
```

```
                    "$$": {
                         "C": [{
                              "$": {
                                   "a": "111",
                                   "b": "222",
                                   "c": "333",
                                   "d": "444"
                              }
                         }, {
                              "$": {
                                   "a": "aaa",
                                   "b": "bbb",
                                   "c": "ccc",
                                   "d": "ddd"
                              }
                         }]
                    }
               }, {
                    "$": {
                         "x": "ccc",
                         "C": "ffg",
                         "y": "vvv",
                         "z": "bbb"
                    },
                    "$$": {
                         "C": [{
                              "$": {
                                   "q": "qqq",
                                   "w": "www",
                                   "c": "ccc",
                                   "r": "rrr"
                              }
                         }]
                    }
               }]
          }
     }]
  }
}
```

Формат json-6

Атрибуты и узлы с одинаковыми именами объединяются, атрибут идет первым.

```
{
    "A": [{
      "b": "attr_b",
      "B": ["attr_B", {
           "C": ["attr_c", {
                "a": "111",
                "b": "222",
                "c": "333",
                "d": "444"
```

```
            }, {
                 "a": "aaa",
                 "b": "bbb",
                 "c": "ccc",
                 "d": "ddd"
            }]
      }, {
            "x": "ccc",
            "y": "vvv",
            "z": "bbb",
            "C": ["ffg", {
                 "q": "qqq",
                 "w": "www",
                 "c": "ccc",
                 "r": "rrr"
            }]
      }]
    }]
}
```

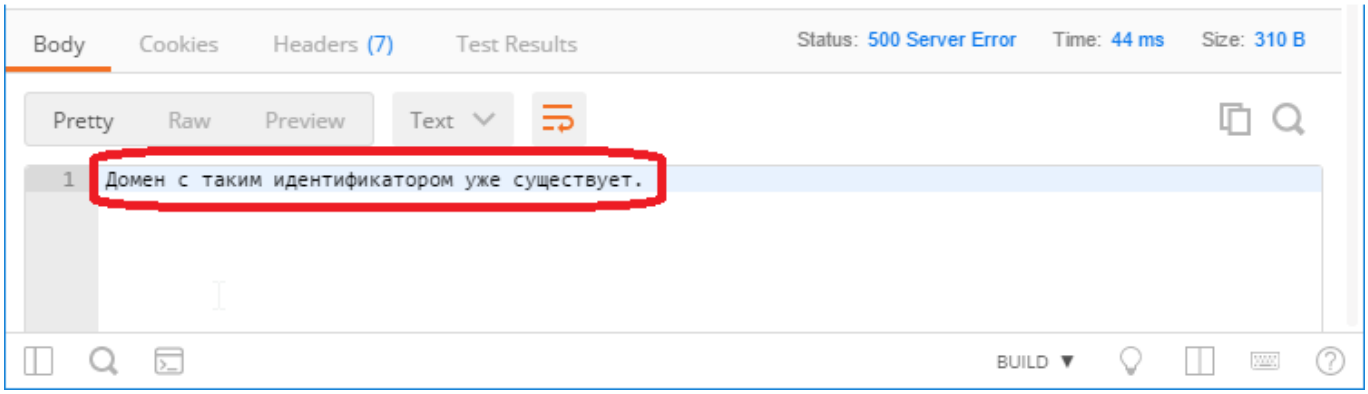
Формат json-7 или json (используется по умолчанию)

Перед именем узла всегда ставится префикс **$** (символ доллара).

```
{
    "$A": [{
      "B": "attr_B",
      "b": "attr_b",
      "$B": [{
            "C": "attr_c",
            "$C": [{
                 "a": "111",
                 "b": "222",
                 "c": "333",
                 "d": "444"
            }, {
                 "a": "aaa",
                 "b": "bbb",
                 "c": "ccc",
                 "d": "ddd"
            }]
      }, {
            "x": "ccc",
            "C": "ffg",
            "y": "vvv",
            "z": "bbb",
            "$C": [{
                 "q": "qqq",
                 "w": "www",
                 "c": "ccc",
                 "r": "rrr"
            }]
      }]
```

```
    }]
}
```

В Web-клиенте и в программе Postman при возникновении ошибки, команда возвращает строку с описанием возникшей ошибки.



Формат XML-документов в дополнительном параметре

Многие команды (например, *create_class*) требуют для своей работы данные структурированные в виде XML-документов. Эти данные передаются команде в дополнительном параметре. XML-документ может быть представлен как в стандартном XML-формате, так и в одном из JSON-форматов. Описание JSON-форматов приведено в статье «Возвращаемое значение в Web-клиенте».

Команда автоматически распознает формат данных. В примерах, приведенных для команд, входные данные представлены в XML-формате, поскольку все элементы базы данных хранятся в этом формате и просмотр их кода средствами ODANT осуществляется в этом формате.

Внимание! Параметр *format*, описанный выше, определяет формат данных, выдаваемых командой, и не влияет на распознавание формата входных данных.

Ниже в качестве примера приведено описание класса в XML- и JSON-форматах:

```
<?xml version="1.0"?>
    <CLASS ClassId="1D0F54787BAA8B4" Name="CreatedClass">
        <METADATA>
            <ATTR Name="Город" Label="Город" List="False"/>
            <ATTR Name="Улиц" Label="Улица" List="False"/>
            <ATTR Name="Дом" Label="Дом" List="False"/>
            <ATTR Name="Квар" Label="Квартира" List="False"/>
        </METADATA>
    </CLASS>
```

```
{
   "$CLASS": [{
      "ClassId": "1D0F54787BAA8B4",
      "Name": "CreatedClass",
      "$METADATA": [{
         "$ATTR": [
            {
            "Name": "Город",
            "Label": "Город",
```

```
            "List": "False"
            },
            {
            "Name": "Улиц",
            "Label": "Улица",
            "List": "False"
            },
            {
            "Name": "Дом",
            "Label": "Дом",
            "List": "False"
            },
            {
            "Name": "Квар",
            "Label": "Квартира",
            "List": "False"
            }
         ]
      }]
   }]
}
```

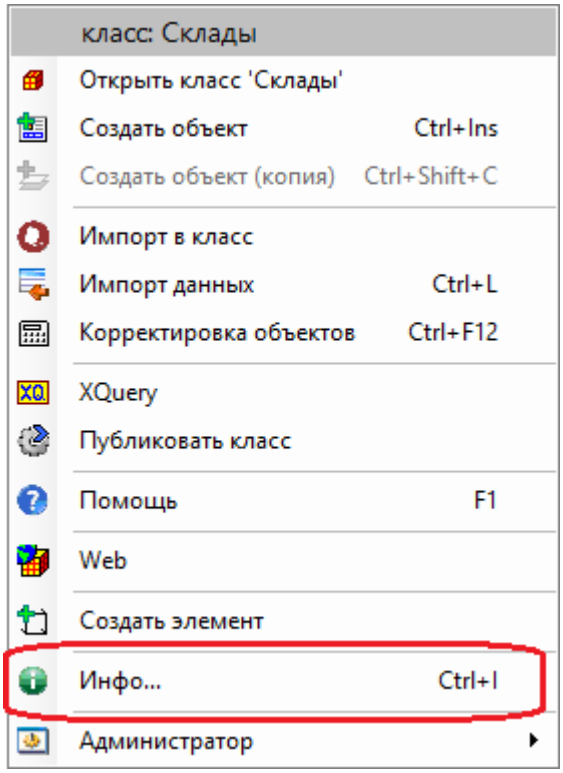
Параметр ssid

Команды имеют необязательный параметр ***ssid***. Значением этого параметра является идентификатор сессии пользователя. Идентификатор сессии пользователя используется для авторизации в Web-клиенте. Он формируется на основе идентификатора пользователя, идентификатора сеанса работы и IP-адреса компьютера.

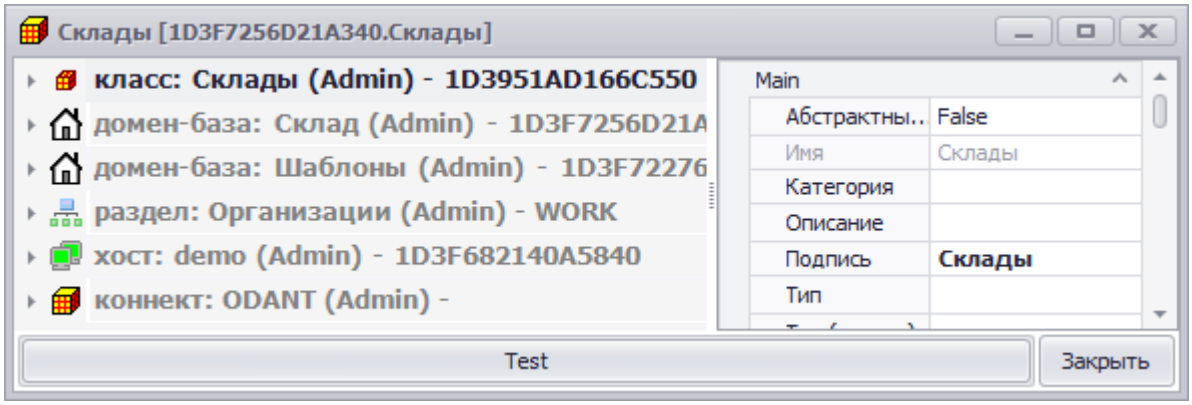
Если параметр ***ssid*** не указан, команда автоматически использует текущий идентификатор сессии пользователя. Используя параметр ***ssid*** можно выполнить команду от имени другого пользователя (точнее от имени его сессии).

Узнать текущий идентификатор сессии пользователя можно в Windows-клиенте следующим образом:

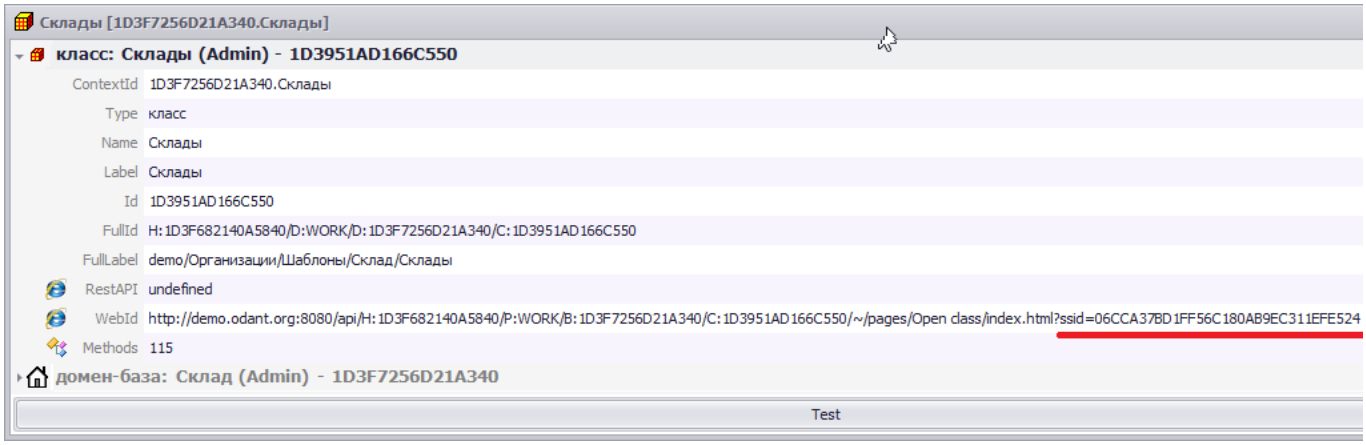
1) Выберите в проводнике любой класс и щелкните по нему ПКМ. Откроется контекстное меню класса;
2) В контекстном меню выберите пункт «***Инфо...***»:



3) Откроется форма с информацией о классе:



4) Раскройте ветку «***класс***». Идентификатор сессии пользователя отображается в конце поля ***WebId*** (на рисунке идентификатор подчеркнут красной чертой):



1 Команды работы с асинхронными процессами

1.1 get_async_result

Выражение

```
string get_async_result( number asyncid )
```

Rest API

```
SourceWebID?method=get_async_result&asyncid=AsyncCommandID
```

Описание

Команда ожидает окончания выполнения и выдает результат выполнения асинхронного процесса.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
asyncid	число	Идентификатор асинхронного процесса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Результат выполнения асинхронного процесса.

Пример использования

В связи с большой трудоемкостью и неактуальностью использования данной команды в командной строке Windows-клиента и Web-браузера, примеры использования команды не приводятся.

1.2 get_async_progress

Выражение

```
number get_async_progress( number asyncid )
```

Rest API

```
SourceWebID?method=get_async_progress&asyncid=AsyncCommandID
```

Описание

Выдает процент выполнения асинхронного процесса (отрицательное значение - идентификатор асинхронного процесса не распознан).

Примечание – Текущая версия команды возвращает два значения:

0 – если процесс не завершился;

100 – если процесс завершился.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
asyncid	число	Идентификатор асинхронного процесса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: число

Описание: Процент полученного результата асинхронной команды.

Пример использования

В связи с большой трудоемкостью и неактуальностью использования данной команды в командной строке Windows-клиента и Web-браузера, примеры использования команды не приводятся.

2 Команды работы с доменами

2.1 create_domain

Выражение

```
string create_domain(string id, string name)
```

Rest API

```
SourceWebID?method=create_domain&name=NewDomainName
```

Описание

Команда создает дочерний домен в указанном родительском домене. Команда позволяет создать:

— домен без типа. Для этого достаточно в параметре команды *name* указать имя создаваемого домена, а дополнительный параметр не используется. Параметр *name* можно опустить. В этом случае программа сама сгенерирует имя для домена. Имя будет состоять из слова *Domain* и идентификатора домена, разделенных пробелом (например, *Domain 1D3DE1CE69E0934*);

— домен на основании XML-шаблона. Шаблон передается команде через дополнительный параметр. При наличии дополнительного параметра в команде, параметр *name* игнорируется, а имя класса задается в XML-шаблоне или генерируется автоматически.

При создании домена из XML-шаблона команда автоматически добавляет в тег *<CLASS>* в XML-описании домена атрибуты *Name*, *ClassId* и *Update*. Если указанные атрибуты присутствуют в XML-шаблоне, то атрибуты *Name* и *ClassId* копируются в XML-описание без изменений, а значение атрибута *Update* заменяется на актуальное значение.

При создании домена без использования XML-шаблона, но с использованием параметра *name*, команда создает следующее XML-описание домена:

```
<?xml version="1.0"?>
      <CLASS Name="MyDomain" ClassId="1D408A8C896F511" Label="MyDomain"/>
```

Примечание – Атрибуты *Name* и *Label* всегда имеют значение, переданное команде в параметре *name*. Конкретные значения остальных атрибутов зависят от контекста, в котором выполнена команда.

При создании домена без использования XML-шаблона и без использования параметра *name*, команда создает следующее XML-описание домена:

```
<?xml version="1.0"?>
      <CLASS Name="Domain 1D408AA983561F9" ClassId="1D408AA983561F9" Label="Domain
1D408AA983561F9"/>
```

Примечание – Правила автоматического формирования значений атрибутов *Name* и

Label изложены выше. Конкретные значения остальных атрибутов зависят от контекста, в котором выполнена команда.

Команда выдает сообщение об ошибке при попытке создать домен в корне хоста, в системных доменах или в домене-рабочее место.

Примечания

1) Команда не проверяет соответствие типов разделов и типов, создаваемых в них доменов. Например, команда может создать домен-разработчика в разделе *Организации*, хотя визуальная среда разработки не позволяет этого делать.

2) В текущей версии платформы ODANT, команда позволяет создать в одном разделе, но в разных родительских доменах, дочерние домены с одинаковыми идентификаторами, хотя это запрещено принятой в ODANT системой адресации. Результаты операций с такими дочерними доменами непредсказуемы. Поэтому будьте внимательны при использовании в XML-шаблоне атрибута *ClassId*, который прямо задает идентификатор создаваемого домена.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
name	строка	Имя создаваемого домена.
id	строка	Полный идентификатор домена или раздела, в котором создается дочерний домен.

Дополнительный параметр

XML-шаблон создаваемого домена.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – строка	**Тип** – определяется параметром *format*

Полный идентификатор созданного домена.	Структура с описанием атрибутов созданного домена:
	`{`
	`"i":"DomainID",`
	`"n":"DomainName",`
	`"a":"AccessLevel"`
	`}`
	где `i`, `n`, `a` — фиксированные ключи identifier, name и access level;
	`DomainID` — краткий идентификатор созданного домена;
	`DomainName` — имя созданного домена;
	`AccessLevel` — уровень доступа, с которым была выполнена

	команда. Также JSON-структура может содержать другие ключи, соответствующие атрибутам в теге <*CLASS*> в XML-описании созданного домена (например, ключ «t», который соответствует атрибуту *Type*).

Пример №1 команды в Windows-клиенте

Команда

```
ООО Солнышко [Мой компьютер] -> create_domain?name=MyDomain (187 мс)
```

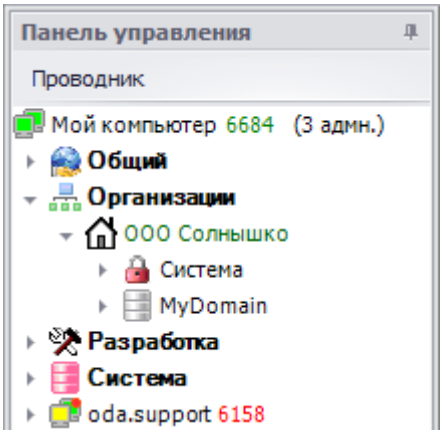
Дополнительный параметр

```
Не используется
```

Результат выполнения

```
/H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/D:1D16FC2679C4810
```

Примечание – Команда создала дочерний домен без типа с именем *MyDomain* в родительском домене-базе *ООО Солнышко*. Созданному домену автоматически присвоен идентификатор *1D16FC2679C4810*.



Пример №2 команды в Windows-клиенте

Команда

```
ООО Солнышко [Мой компьютер] -> create_domain (358 мс)
```

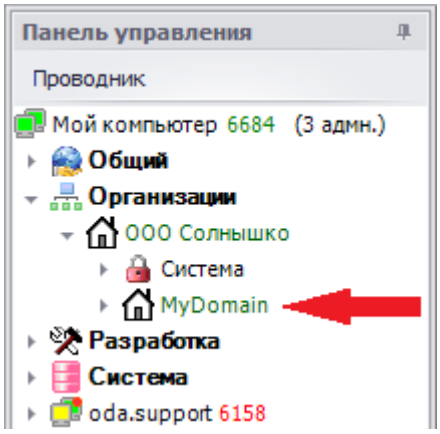
Дополнительный параметр

```
<?xml version="1.0"?>
    <CLASS Name="MyDomain" ClassId="1D0F538F5B31E17" Type="BASE"/>
```

Результат выполнения

```
/H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E17
```

Примечание – Команда создала домен-базу с именем *MyDomain* и идентификатором *1D0F538F5B31E17* в родительском домене-базе *ООО Солнышко*.



Пример №3 команды в Web-браузере

Команда

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6?method=create_domain
&name=MyDomain
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
{"i":"1D3DEFD921D3C30","n":"MyDomain","a":"6"}
```

Примечание – Команда создала дочерний домен без типа с именем *MyDomain* в родительском домене-базе *t_test*. Созданному домену автоматически присвоен идентификатор *1D3DEFD921D3C30*.

Пример №4 команды в программе Postman

Команда

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6?method=create_domain
```

Дополнительный параметр

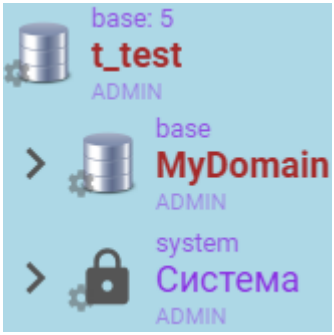
```
<?xml version="1.0"?>
    <CLASS Name="MyDomain" ClassId="1D0F538F5B31E17" Type="BASE"/>
```

Результат выполнения

```
{
    "i": "1D0F538F5B31E17",
    "n": "MyDomain",
    "t": "BASE",
```

```
    "a": "6"
}
```

Примечание – Команда создает домен-базу с именем *MyDomain* и идентификатором *1D0F538F5B31E17* в родительском домене-базе *t_test*.



2.2 delete_domain

Выражение

```
string delete_domain( string id )
```

Rest API

```
SourceWebID?method=delete_domain
```

Описание

Команда удаляет указанный домен из структуры базы данных.

Примечания

1) Подкаталог с доменом не удаляется из каталога родительского домена, а перемещается в подкаталог «removed», расположенный в каталоге родительского домена.

2) Если команде передать полный идентификатор класса или объекта класса, то вместе с указанным классом будет удален его родительский домен.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор удаляемого домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Windows-клиент	Web-клиент

Тип – строка	**Тип** – определяется параметром *format*

Краткий идентификатор удаленного домена.	{
	"result":"DomainID"
	}
	где `result` — фиксированный ключ;
	`DomainID` — краткий идентификатор удаленного домена.

Пример команды в Windows-клиенте

Команда:

```
Касса [Мой компьютер/ООО Солныш.] -> delete_domain (62 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
1D16ED61C49E780
```

Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D0F538F5B31E17?method=delete_domain
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
{"result":"1D0F538F5B31E17"}
```

2.3 get_child_domains

Выражение

```
string get_child_domains( string id )
```

Rest API

В текущей версии платформы ODANT данная команда реализована только для Windows-клиента.

Описание

Команда выдает список полных идентификаторов доменов, являющихся дочерними (непосредственными потомками) по отношению к домену, указанному в параметре ***id***. В списке присутствуют только идентификаторы доменов, к которым текущий пользователь имеет доступ. Эти домены пользователь может видеть в проводнике. Команду можно применять к доменам любого типа, включая разделы.

Примечание – В текущей версии Windows-клиента команда включает в список дочерних доменов раздела «Общий» идентификаторы доменов-модулей, к которым пользователь не имеет доступа и которые, соответственно, не отображаются в проводнике. Такая же проблема с доменами-организациями. Например, если пользователь имеет доступ к домену-рабочему месту, расположенному в домене-организации, то в списке дочерних доменов помимо этого рабочего места будут также присутствовать все домены-модули, находящиеся в этом домене-организации, в тоже время дочерние домены других типов в список не включаются.

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор родительского домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Список полных идентификаторов дочерних доменов. Идентификаторы в списке разделяются символом ";" (точка с запятой). Если домен не имеет дочерних доменов, команда возвращает сообщение *no result*.

Пример команды в Windows-клиенте

Команда:

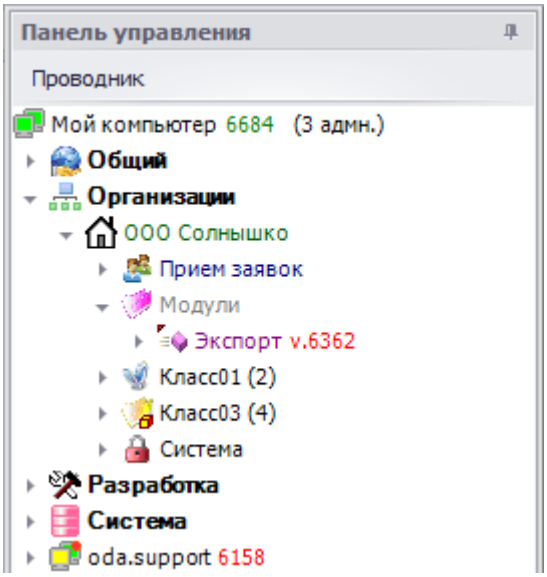
```
ООО Солнышко [Мой компьютер] -> get_child_domains (12 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
/H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/D:1CF8163B948EE86;/H:1D0EB94C5ACD42E/D:WO
RK/D:1D0F538F5B31E18/D:1D0F5FCD4EEE664;/H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/D:
000000000000000
```



Примечание - На локальном хосте в домен-базу *ООО Солнышко* входят:

- домен-рабочее место *Прием заявок* с идентификатором 1D0F5FCD4EEE664;

- домен-модуль *Экспорт* с идентификатором 1CF8163B948EE86;

- системный домен *Система* с идентификатором 000000000000000.

2.4 get_domain

Выражение

```
string get_domain(string id, string domain)
```

Rest API

```
SourceWebID?method=get_domain&domain=DomainID|DomainName
```

Описание

Команда проверяет наличие дочернего домена с идентификатором или именем, переданным в параметре *domain*, в родительском домене, переданном в параметре *id*.

Примечания

1) Команда позволяет проверять наличие дочернего домена в родительском, даже если оба этих домена недоступны текущему пользователю, но известны идентификаторы этих доменов.

2) В текущей версии платформы ODANT команда находит дочерний домен с именем *System* в домене любого типа. В место идентификатора домена команда возвращает строку «SYSTEM».

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор родительского домена или раздела.
domain	строка	Краткий идентификатор или имя проверяемого дочернего домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – строка	**Тип** – определяется параметром *format*

Идентификатор дочернего домена, если дочерний домен существует.	{
	"result":"DomainID"
	}
	где result — фиксированный ключ;
	DomainID — краткий идентификатор проверяемого домена.

Если искомый дочерний домен не существует, команда возвращает пустую строку.	

Пример команды в Windows-клиенте

Команда:

```
1.Платформа [oda.support/ODANT] -> get_domain?domain=1D2723C632492F0 (1 мс)
```

или

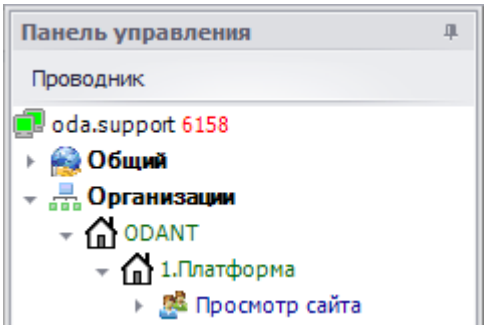
```
1.Платформа [oda.support/ODANT] -> get_domain?domain=Просмотр сайта (1 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
1D2723C632492F0
```



Примечание – На хосте *oda.support* домен-база *1.Платформа* содержит домен-рабочее место *Просмотр сайта* с идентификатором *1D2723C632492F0*.

Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6?method=get_domain&domain=MyDomain
```

или

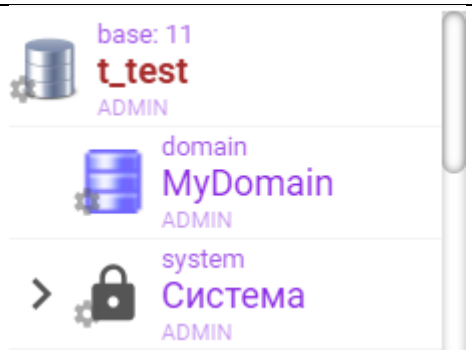
```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6?method=get_domain&domain=1D0F538F5B31E22
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
{"result":"1D0F538F5B31E22"}
```



Примечание – На хосте *odant.org* домен-база *t_test* содержит домен *MyDomain* с идентификатором *1D0F538F5B31E22*.

2.5 get_domain_state

Выражение

```
number get_domain_state(string id)
```

Rest API

```
SourceWebID?method=get_domain_state
```

Описание

Команда выдает совокупное состояние домена в виде комбинации флагов:

1 – имеются сервисы;

2 – имеются классы;

4 – есть дочерние домены с сервисами;

8 – есть дочерние домены с классами;

16 – админ текущего домена;

32 – админ в дочернем домене.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: Число

Описание: Совокупное состояние домена в виде комбинации флагов.

Windows-клиент	Web-клиент
Тип – число	**Тип** – определяется параметром *format*

Совокупное состояние домена в виде комбинации флагов.	{
	"result":"Flags"
	}
	где result — фиксированный ключ;
	Flags — комбинация флагов состояния домена.

Пример команды в Windows-клиенте

Команда:

```
Демо [oda.support] -> get_domain_state (0 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
3
```

Примечание - Домен-организация *Демо* на хосте *oda.support* содержит сервисы и классы.

Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6?method=get_domain_state
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
{"result":"12"}
```

2.6 is_domain_admin

Выражение

```
bool is_domain_admin(string id)
```

Rest API

```
SourceWebID?method=is_domain_admin
```

Описание

Команда проверяет, является ли текущий пользователь администратором указанного домена.

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – логический	**Тип** – определяется параметром *format*

`true` – если текущий пользователь является администратором указанного домена;
`пустая строка` – если текущий пользователь не является администратором указанного домена.
	Если текущий пользователь является администратором указанного домена:
	`{`
	`"result":"true"`
	`}`
	
	Если текущий пользователь не является администратором указанного домена:
	`пустая строка`

Пример команды в Windows-клиенте

Команда:

```
ООО Солнышко [Мой компьютер] -> is_domain_admin (0 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
true
```

Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6?method=is_domain_admin
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

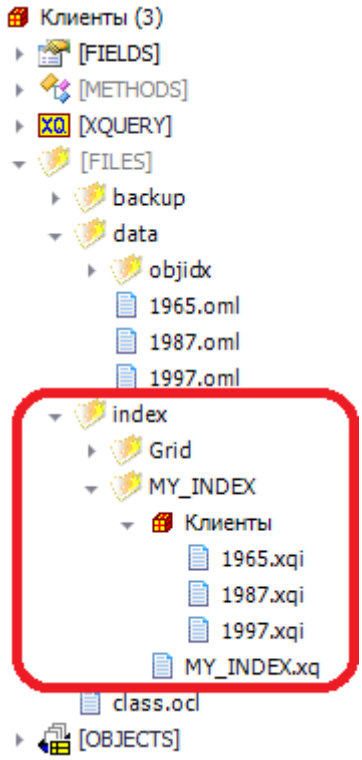
```
{"result":"true"}
```

3 Команды работы с индексами

Индексы представляют собой XML-документы, содержащие упрощенное описание объектов класса. Индексы предназначены для ускорения тех или иных операций с классами и объектами в базе данных.

Все индексы класса и тексты запросов на создание индексов хранятся в файловой структуре класса в папке *index*. В этой папке для каждого индекса создается отдельный каталог с именем индекса (например, *MY_INDEX*). В этом каталоге создается файл с именем индекса и расширением **xq** (например, *MY_INDEX.xq*), который содержит текст запроса. Кроме того, в этом каталоге создается подкаталог, именем которого является 15-разрядный идентификатор класса. В этом подкаталоге находятся файлы с индексами. Файлы с индексами имеют расширение **xqi**. Имена файлов с индексами совпадают с именами пакетов с объектами, расположенными в папке *data*, т.е. каждому пакету соответствует свой индекс.

Пример файловой структуры класса с индексами:

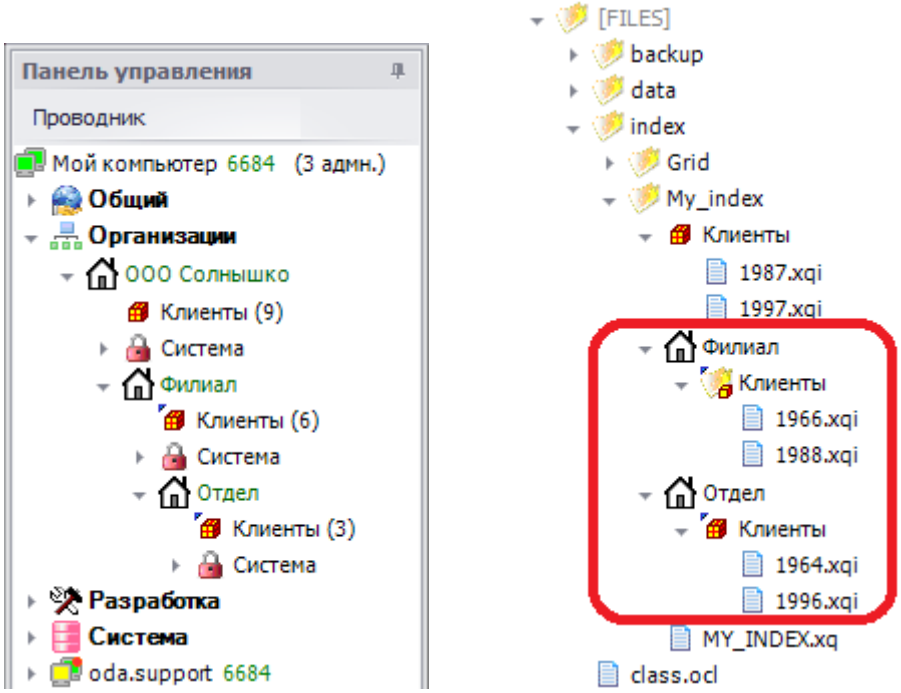


Пояснения к примеру:

1) В классе установлен режим пакетирования по году рождения клиента:
2) Индекс *Grid* автоматически создается во всех классах и содержит данные для стандартного представления объектов в списке объектов.
3) В примере приведена файловая структура класса, отображаемая в конфигураторе. На рисунке подкаталог с файлами индексов называется *Клиенты*, хотя в файловой системе на диске подкаталог имеет имя состоящие из 15-разрядного идентификатора класса. Это происходит потому, что конфигуратор всегда заменяет идентификаторы классов в именах файлов и каталогов именами классов.

Рассмотрим случай, когда домен-база имеет дочерние домены-базы. Пусть родительский и дочерние домены-базы содержат классы с одинаковыми идентификаторами. В этом случае класс, расположенный в родительском домене, будет являться родительским для классов в дочерних доменах (режим наследования по идентификатору). В родительском классе будут отображаться все объекты из дочерних классов. Запросы на создание индексов в родительском классе также создают индексы из объектов дочерних классов. Индексы из объектов дочерних классов хранятся в файловой структуре родительского класса. Для хранения этих индексов, в каталоге с именем соответствующего индекса создается отдельный подкаталог для каждого дочернего класса. Именем этих подкаталогов является 15-разрядный идентификатор дочернего домена, в котором расположен дочерний класс. Идентификатор в имени подкаталога имеет префикс «**d.**» (пример имени подкаталога: «d.1D1234567890ABC»). В каждом подкаталоге расположена папка, именем которой является 15-разрядный идентификатор класса. В этих папках расположены файлы с индексами дочерних классов. В родительском классе хранятся индексы всех классов-потомков независимо от уровня вложенности доменов-баз, в которых классы-потомки расположены.

Пример файловой структуры родительского класса с индексами:



Пояснения к примеру:

1) На левом рисунке представлена структура базы данных *ООО Солнышко*, имеющая дочерние домены-базы *Филиал* и *Отдел*. В составе базы данных *ООО Солнышко*, находится класс *Клиенты*, который имеет дочерние классы в доменах-базах *Филиал* и *Отдел*.
2) На правом рисунке представлена файловая структура класса. Структура папки *My_index* развернута для демонстрации принципа хранения индексов из дочерних классов. Как видно из рисунка, отношения подчиненности между дочерними доменами не сохраняются в файловой структуре индексов, все подкаталоги с индексами от дочерних классов находятся на одном уровне вложенности.
3) В примере приведена файловая структура класса, отображаемая в конфигураторе. Поэтому вместо реальных имен папок, состоящих из

идентификаторов классов и доменов, отображаются имена соответствующих классов и доменов.

Дочерние классы не наследуют от родительского класса запросы на создание индексов.

Команды работы с индексами не различают строчные и заглавные буквы в имени индекса.

Внимание!!! При изменении режима пакетирования объектов, папка *index* удаляется полностью из файловой структуры класса. После изменения режима пакетирования необходимо заново восстанавливать запросы на создание индексов и сами индексы.

При удалении класса из структуры базы данных, файловая структура класса не удаляется с физического носителя, а переносится в подкаталог «*.removed*», расположенный в каталоге родительского класса или домена. При этом папка *index* удаляется полностью из файловой структуры класса. Поэтому необходимо помнить, что при восстановлении удаленного класса, также необходимо заново восстанавливать запросы на создание индексов и сами индексы.

Адресация индексов

Полный адрес индекса состоит из полного идентификатора класса, после которого указывается имя индекса с префиксом «**I:**» (заглавная латинская буква **I** с двоеточием). Идентификатор класса отделяется от имени индекса символом «**/**» (прямой слеш). Пример адреса:

H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3**/I:MY_INDEX**

Регистр букв в имени индекса игнорируется.

Многоуровневые индексы

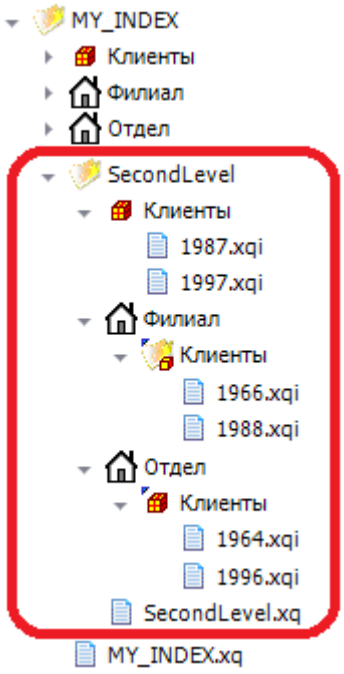
Выше был рассмотрен одноуровневый индекс, формируемый непосредственно из объектов класса. Однако платформа ODANT позволяет создавать сложные многоуровневые структуры из индексов. В такой структуре только первый уровень индексов создается из объектов класса, индексы 2-го и последующих уровней строятся на основе индексов предыдущего уровня. Т.е. запросы на создание индексов первого уровня применяются к объектам класса, а запросы на создание индексов 2-го и последующих уровней применяются к индексам предыдущего уровня.

Многоуровневые структуры индексов имеют древовидную структуру, поэтому при адресации индекса нижнего уровня необходимо указывать все предшествующие индексы более высокого уровня. Пример:

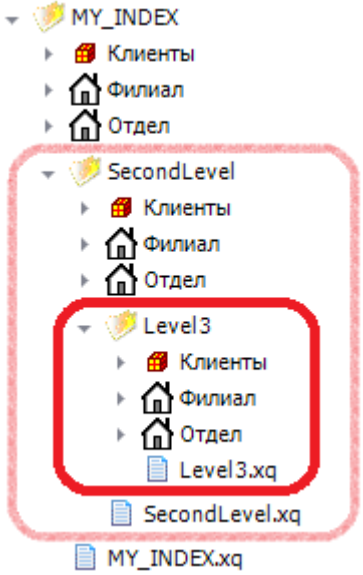
H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3**/I:MY_INDEX\SecondLevel**

Имена индексов разного уровня разделяются в адресе символом «****» (обратный слеш).

Файловая структура индексов второго и последующего уровней располагается в файловой структуре предыдущего уровня и формируется по тем же правилам.



Пример файловой структуры трехуровневого индекса:



Пример XQuery-запросов для многоуровневого индекса

Класс в примере хранит информацию о сотрудниках. Информация включает в себя фамилию, имя, отчество и год рождения сотрудника. Для простоты примера класс содержит одного сотрудника, т.е. класс содержит один объект. Объект хранится в классе в таком виде:

```
<?xml version="1.0"?>
      <PACK name="1987" Name="1987.oml" Date="2018-06-07T12:10:27">
            <OBJECT oid="1D3FE3F5BE28516" Author="1D0EB94C5ACD42E"
Owner="1D0EB94C5ACD42E" cid="1D3E8669A067B9E" bid="1D0F538F5B31E18" version="2"
date="2018-06-07T12:10:20" cnm="Сотрудники" size="315" name="Рыбаков"
Фамилия="Рыбаков" Имя="Иван" Отчество="Петрович" update="2018-06-07T12:10:23"
user="1D0EB94C5ACD42E" Год="1987"/>
      </PACK>
```

Запрос первого уровня выбирает только фамилию, имя и отчество сотрудника. Год рождения нас не интересует:

```
element FirstLevel {
    for $a in //PACK/OBJECT
    return
        element FIO {
            $a/(@oid, @Фамилия, @Имя, @Отчество)
        }
}
```

Индекс первого уровня будет выглядеть так:

```
<?xml version="1.0"?>
      <FirstLevel cid="1D3E8669A067B9E" bid="1D0F538F5B31E18" pack="1987">
            <FIO oid="1D3FE3F5BE28516" Фамилия="Рыбаков" Имя="Иван"
Отчество="Петрович"/>
      </FirstLevel>
```

Запрос второго уровня удаляет отчество сотрудника из индекса:

```
element SecondLevel {
    for $a in //FirstLevel/FIO
    return
        element FI {
            $a/(@oid, @Фамилия, @Имя)
        }
}
```

Индекс второго уровня будет выглядеть так:

```
<?xml version="1.0"?>
      <SecondLevel cid="1D3E8669A067B9E" bid="1D0F538F5B31E18" pack="1987">
            <FI oid="1D3FE3F5BE28516" Фамилия="Рыбаков" Имя="Иван"/>
      </SecondLevel>
```

Запрос третьего уровня оставляет в индексе только фамилию:

```
element ThirdLevel {
    for $a in //SecondLevel/FI
    return
        element F {
            $a/(@oid, @Фамилия)
        }
}
```

Индекс третьего уровня будет выглядеть так:

```
<?xml version="1.0"?>
      <ThirdLevel cid="1D3E8669A067B9E" bid="1D0F538F5B31E18" pack="1987">
            <F oid="1D3FE3F5BE28516" Фамилия="Рыбаков"/>
      </ThirdLevel>
```

Структура индекса

Индекс представляет собой XML-документ, в котором собраны только требуемые данные из объектов класса. Состав и представление данных в индексе определяется запросом на создание индекса.

Для корректного выполнения всех команд API сервера, работающих с индексами, индексы должны иметь двухуровневую структуру хранения информации об объектах. На первом уровне находится корневой элемент, атрибуты которого содержат общие

сведения об индексе. На втором уровне находятся элементы, соответствующие объектам. Каждому элементу второго уровня соответствует один объект класса. Тег, соответствующий корневому элементу, и теги, соответствующие объектам, могут иметь любые имена, главное соблюдать двухуровневую структуру.

В корневом теге автоматически создаются следующие атрибуты:

- cid – идентификатор класса, к которому относится данный индекс;
- bid – идентификатор домена-базы, в котором расположен класс с данным индексом;
- pack – имя пакета, к которому относится данный индекс.

Перечисленные атрибуты очень важны для правильной работы механизма индексов и создаются ядром платформы ODANT автоматически, поэтому запрос на создание индексов не должен создавать эти атрибуты.

Для корректного выполнения всех команд API сервера, работающих с индексами, теги второго уровня должны содержать атрибут *oid*, с кратким идентификатором соответствующего объекта. Этот атрибут не создается автоматически, операторы для его создания должны быть явно заданы в запросе. Идентификаторы объектов хранятся в XML-описании объекта в системном атрибуте *oid* в теге *OBJECT* (см. примеры запросов ниже в текущем разделе).

Пример того, как должен выглядеть типичный индекс, вы можете найти в папке *index\Grid* любого класса. В этой папке содержится индекс для стандартного представления списка объектов.

Служебные индексы *Grid*, *Table*, *Pack, Names* и *~Search~*

В ядре ODANT предусмотрено автоматическое построение некоторых индексов, используемых для служебных целей. Пользователю достаточно указать имя служебного индекса в качестве контекста команды, а XQuery-запросы для этих индексов и сами индексы будут сформированы ядром ODANT автоматически.

Индекс **Grid** используется для типового представления списка объектов в Windows-клиенте. XQuery-запрос и сам индекс формируются автоматически при первом открытии списка объектов класса. При всех последующих изменениях в структуре класса XQuery-запрос и индекс автоматически перестраиваются.

Индекс **Table** аналогичен индексу *Grid*, но используется для отображения типового списка объектов в Web-клиенте.

Индекс **Pack** содержит всю информацию из пакетов данных. Используется если необходимо работать с пакетами данных как с индексами.

Индекс **Names** содержит только имена объектов.

Индекс **~Search~** используется в качестве промежуточных данных в операциях поиска. Создается автоматически командой search_index. Этот индекс содержит полное XML-описание объектов в упрощенном формате. Для каждого объекта в индексе сохраняется только атрибут *oid* с идентификатором объекта и создается новый атрибут **V**, в котором сохраняется содержимое всех атрибутов объекта, объединенное в одну строку. Значения атрибутов в этой строке разделены символом пробела. Все буквы в строке приводятся к нижнему регистру. Объединение значений атрибутов в одну строку упрощает и ускоряет процесс поиска заданной подстроки.

Параметр *loadmask*

Большинство команд работы с индексами содержит параметр *loadmask*. Этот параметр используется только для классов, у которых установлен режим распределения объектов по пакетам.

Параметр *loadmask* указывает команде, с какими пакетами объектов или индексов будет работать команда. В этом параметре можно указать имя конкретного пакета или использовать маску для выбора нескольких пакетов. В маске допускается использовать стандартные подстановочные символы для имен файлов: "*****" и "**?**".

Параметр *loadmask* можно не указывать при вызове команды. В этом случае считается, что значением параметра *loadmask* является подстановочный символ "*****", что соответствует разрешению на обработку командой всех пакетов.

Пример:

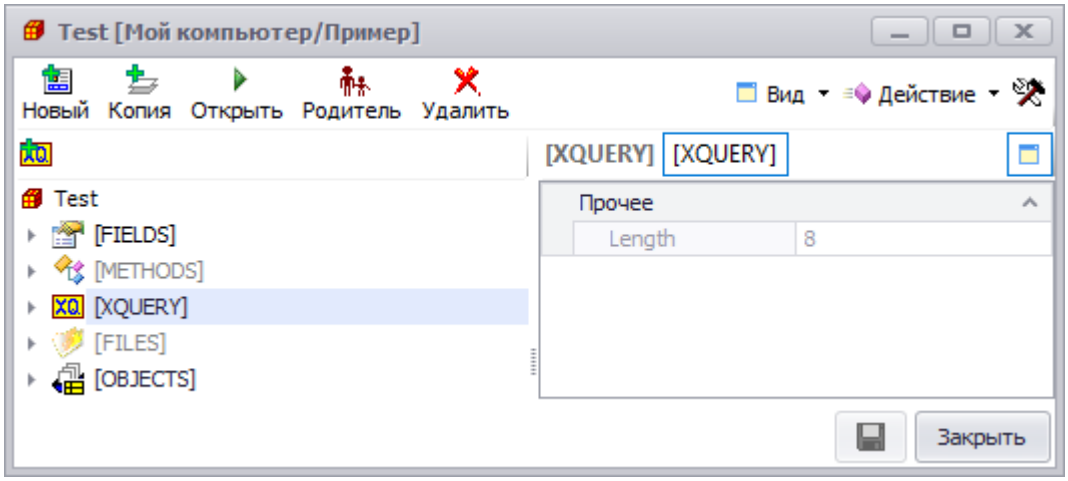
Пусть объекты распределены по пакетам по дате создания, тогда команда с параметром *loadmask=2015** будет работать только с индексами для объектов, созданных в 2015 году.

Включение текста запроса в описание класса

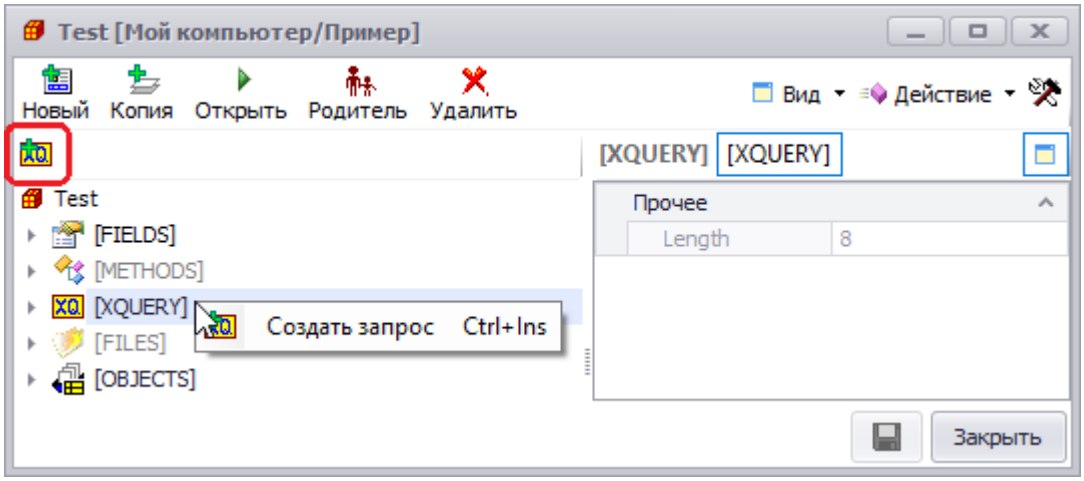
Как уже упоминалось выше, тексты запросов на создание индексов хранятся в файлах с расширением **xq** в соответствующих подкаталогах папки *index*. Текст запроса сохраняется в файле командой *create_index*. Если текст запроса был передан команде в дополнительном параметре, то созданный файл с расширением **xq** оказывается единственным местом хранения текста запроса в базе данных. В процессе настройки базы данных папка *index* может быть автоматически удалена, и тексты запросов могут быть безвозвратно утеряны, поскольку указанные файлы не сохраняются в архиве.

Описанной проблемы можно избежать, если включить текст запросов в XML-описание класса. Это делается следующим образом:

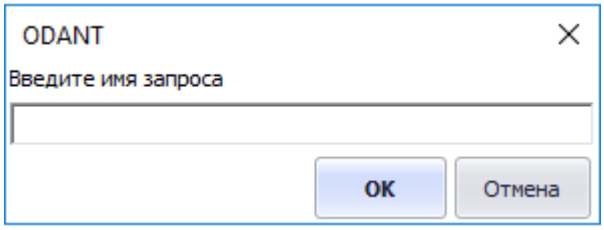
1) Откройте конфигуратор класса и выделите в структуре класса раздел [XQUERY]:



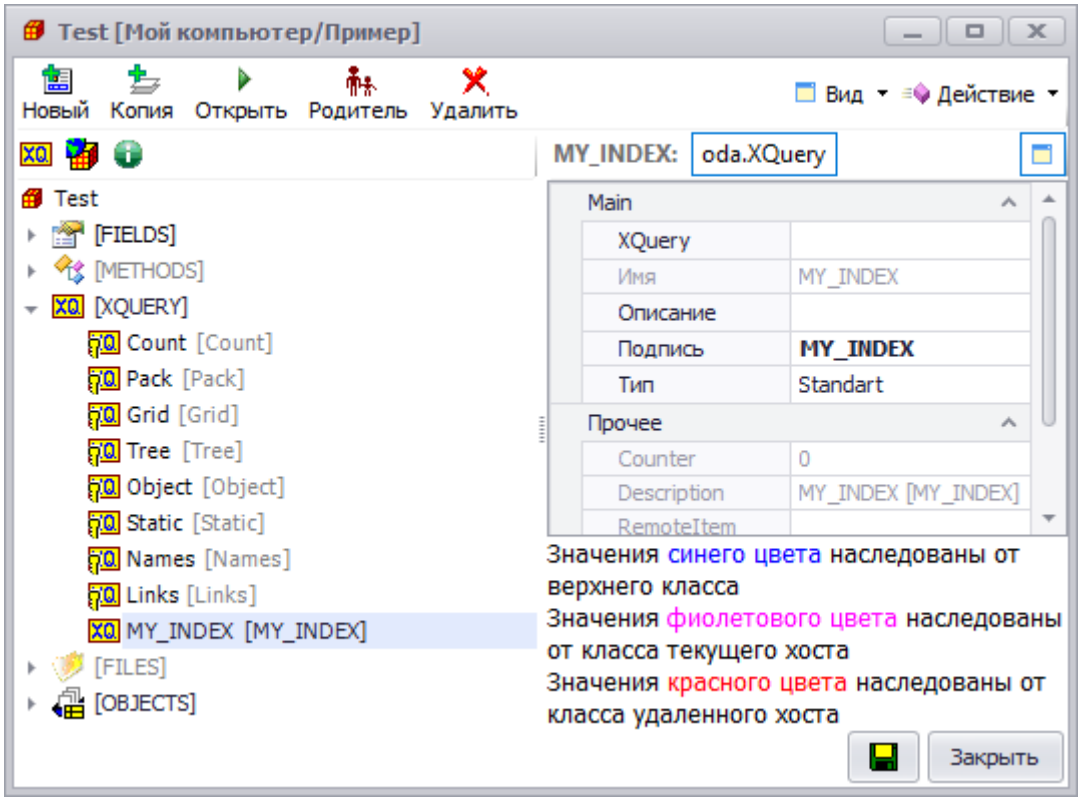
2) Откройте контекстное меню раздела [XQUERY] и выберите пункт «Создать запрос» или нажмите кнопку **XQ** на панели управления:



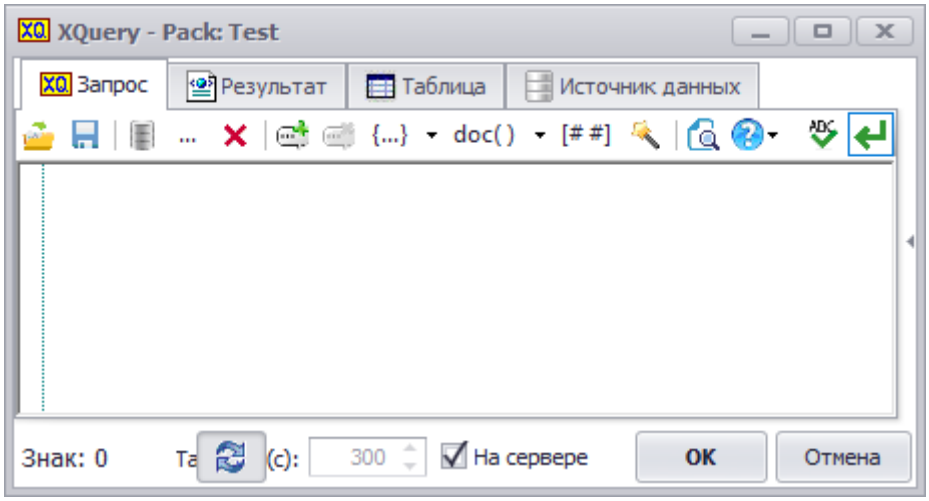
Откроется форма для ввода имени индекса:



3) Введите требуемое имя и нажмите кнопку **OK**. В разделе [XQUERY] появится созданный запрос. В примере созданный запрос имеет имя *MY_INDEX*:



4) Дважды щелкните по имени созданного запроса, откроется редактор для ввода текста запроса:



5) Ведите текст запроса в редактор и сохраните его.

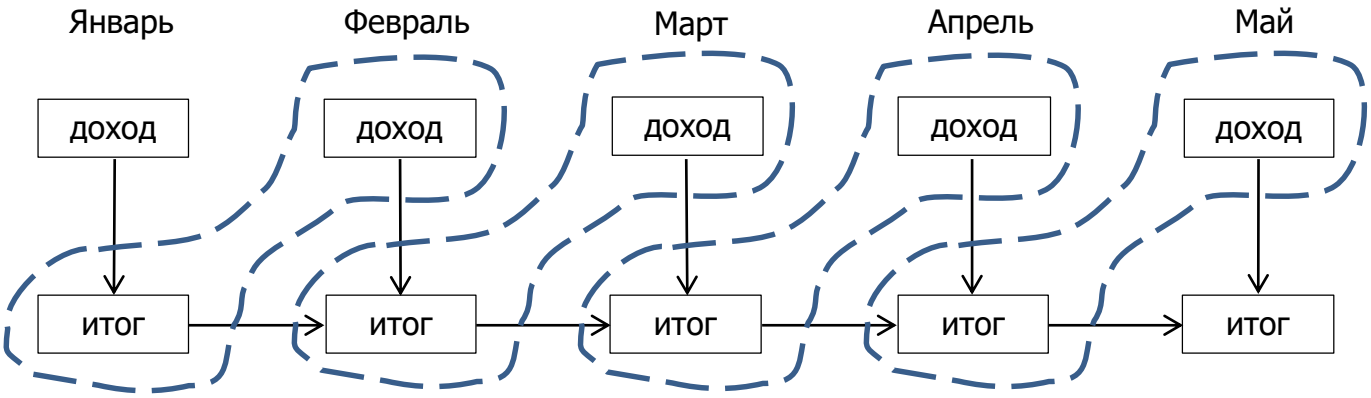
Теперь текст запроса на создание индекса будет храниться в XML-описании класса в файле *class.ocl*. Правила работы с запросами на создание индекса приведены ниже в описании соответствующих команд.

Внимание! Текущая версия платформы позволяет создавать в разделе [XQUERY] конфигуратора XQuery-запросы с одинаковыми именами, различающимися только регистром букв. А при выполнении API-команд работы с индексами регистр букв в именах XQuery-запросов игнорируется. Поэтому при наличии в классе двух запросов с одинаковыми именами, различающимися регистром букв, нельзя предугадать какой запрос будет выполнен. Будьте внимательны при выборе имен XQuery-запросов!

Каскадные индексы

Во многих информационных процессах требуется производить расчеты каких-либо параметров нарастающим итогом за заданный период. В простейшем случае можно написать запрос, который просматривает все документы за указанный период и производит необходимые расчеты. При таком подходе при каждом добавлении или изменении документа производится перерасчет итога с самого начала периода. По мере накопления документов такой перерасчет начинает занимать все больше времени. Чтобы сократить объем расчетов необходимо разбить все документы на пакеты и хранить в каждом пакете нарастающий итог актуальный на дату последнего документа в пакете. В этом случае пересчитывать данные придется только для пакетов, в которых произошли изменения, и последующих пакетов.

ODANT позволяет создавать каскадные индексы, предназначены для расчета нарастающего итога. Схематично работа такого индекса показана на рисунке ниже:



Запросы для создания каскадного индекса имеют двух уровневую структуру. На первом уровне находится обычный запрос, создающий упрощенное описание объектов класса. На втором уровне находится запрос, рассчитывающий нарастающий итог для каждого пакета данных.

3.1 create_index

Выражение

```
string create_index(string id, string loadmask)
```

Rest API

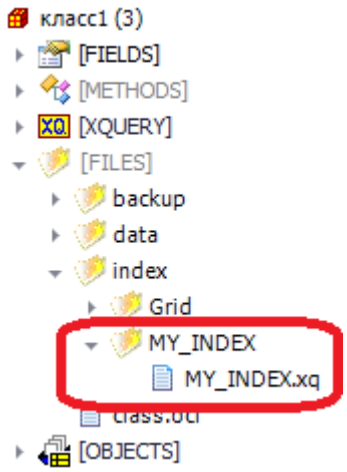
```
SourceWebID?method=create_index&id=ClassFullID/I:IndexName
```

Описание

Команда создает XQuery-запрос для создания индекса класса и заносит его структуру класса. Если файл с одноименным запросом уже существует в классе, то он перезаписывается. Команда создает только XQuery-запрос, а сам индекс в структуре класса не создается. Файлы с индексом можно создать позже командой *update_index*.

Примечание – Сразу после создания XQuery-запрос выполняется и создает в оперативной памяти в кэше соответствующий индекс. Все последующие запросы к этому индексу будут обращаться к кэшу, что ускоряет их выполнение. Но, повторяю, эта команда не создает файлы с индексом в файловой структуре класса.

В результате выполнения команды в файловой структуре класса в папке *index* создается каталог с именем создаваемого индекса. В этом каталоге создается файл с именем индекса и расширением «***xq***», содержащий код запроса. Пример:



Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, для которого создается индекс, с указанием наименования индекса.
loadmask	строка	Маска имен пакетов с объектами, из которых должен быть создан индекс в кэше (см. подробное описание параметра во Команды работы с индексами).

Дополнительный параметр

XQuery-запрос для создания индекса.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – логический	**Тип** – определяется параметром *format*

`true` – команда выполнена успешно;
`сообщение об ошибке`, если XQuery-запрос не создан.
	В случае у спешного выполнения команды:
	`{`
	`"result": "true"`
	`}`
	
	Если XQuery-запрос не создан:
	`сообщение об ошибке`

Пример команды в Windows-клиенте

Команда:

```
ООО Солнышко [Мой компьютер] ->
create_index?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3/I:MY_IND
EX (46 мс)
```

Дополнительный параметр:

```
element x {
    for $a in //PACK/OBJECT
    return
        element O {
            $a/(@oid, @Фамилия, @Имя, @Отчество)
        }
}
```

Результат выполнения:

```
true
```

Примечание – Команда создает в файловой структуре класса в папке *Index* подкаталог *My_index* с файлом с *MY_INDEX.xq*, который содержит XQuery-запрос, переданный через дополнительный параметр.

Пример команды в программе Postman

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3E3AF508F7060?method=create_index&id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3E3AF508F7060/I:MY_INDEX
```

Дополнительный параметр:

```
element x {
    for $a in //PACK/OBJECT
    return
        element O {
            $a/(@oid, @Фамилия, @Имя, @Отчество)
        }
}
```

Результат выполнения:

```
{
    "result": "true"
}
```

Примечание – Команда создает в файловой структуре класса в папке *Index* подкаталог *My_index* с файлом с *MY_INDEX.xq*, который содержит XQuery-запрос, переданный через дополнительный параметр.

3.2 get_class_indexes

Выражение

```
string get_class_indexes(string id)
```

Rest API

```
SourceWebID?method=get_class_indexes
```

Описание

Команда возвращает список индексов указанного класса. Имена индексов в списке разделяются символом "|" (вертикальная черта).

Фактически команда проверяет наличие файлов с запросами на построение индексов, а не наличие самих индексов.

Примечание – Команда возвращает список индексов, даже если класс недоступен текущему пользователю, но известен полный идентификатор этого класса.

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Windows-клиент	**Web-клиент**
Тип – строка	**Тип** – определяется параметром *format*

Имена индексов, разделенные символом "|" (вертикальная черта).	{
	"result":"ListOfIndexes"
	}
	где result — фиксированный ключ;
	ListOfIndexes — имена индексов, разделенные символом "|" (вертикальная черта).

Пример команды в Windows-клиенте

Команда:

```
Класс02 [Мой компьютер/ООО Солныш.] -> get_class_indexes (15 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
Grid|MY_INDEX
```

Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3E3AF508F7060?method=get_class_indexes
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
{"result":"Grid|MY_INDEX"}
```

3.3 search_index

Выражение

```
XML search_index( string id, string loadmask, bool allwords)
```

Rest API

```
SourceWebID?method=search_index&id=ClassFullID/I:IndexName
```

Описание

Команда осуществляет поиск подстроки в значениях атрибутов объектов. Искомая подстрока передается в дополнительном параметре. Поиск осуществляется в XML-описании объектов среди значений всех атрибутов, в том числе и среди системных атрибутов, за исключением атрибутов *cid*, *bid* и *update*. Регистр букв при поиске не учитывается. Если объекты класса разделены на пакеты, то поиск осуществляется только в пакетах, соответствующих маске, указанной в параметре *loadmask*.

В процессе работы команда осуществляет несколько операций:

1) Команда создает в папке *index* специальный служебный индекс с именем *~Search~*. Этот индекс содержит полное XML-описание объектов в упрощенном формате. Для каждого объекта в индексе сохраняется только атрибут *oid* с идентификатором объекта и создается новый атрибут **V**, в котором сохраняется содержимое всех атрибутов объекта, объединенное в одну строку. Значения атрибутов в этой строке разделены символом пробела. Все буквы в строке приводятся к нижнему регистру. Объединение значений атрибутов в одну строку упрощает и ускоряет процесс поиска заданной подстроки. Если индекс *~Search~* уже существует и не произошло изменений объектов класса, то индекс повторно не создается и не обновляется.
2) Команда ищет в индексе *~Search~* все объекты, у которых атрибут **V** содержит заданную подстроку, и запоминает их идентификаторы. Т.е. поиск подстроки происходит не в самом объекте и его атрибутах, а в объединенной строке в индексе *~Search~*.
3) Команда выбирает из индекса, указанного в параметре **id**, все объекты с идентификаторами, отобранными на шаге 2, и выдает эти объекты в качестве результата поиска.

Операция поиска подстроки содержащей пробел имеет следующую особенность. Искомая подстрока как-бы разделена пробелом на две части. А в объединенной строке в индексе *~Search~* значения атрибутов соединены пробелом. При этом возможна ситуация, когда первая часть искомой подстроки будет найдена в одном атрибуте, а вторая часть искомой подстроки будет найдена в следующем за ним атрибуте. Т.е. будет выбран объект, атрибуты которого не содержат искомой подстроки.

Рассмотрим пример ошибочного выбора объекта, не содержащего искомую подстроку. Пусть мы хотим найти в каталоге товар с артикулом «16 2018» (код артикула содержит пробел):

XML-описание товара в индексе *~Search~*:

```
<F oid="1D3E935076E4CB8" V="1d3e935076e4cb8 1d0eb94c5acd42e
1d0eb94c5acd41e 16 2018-05-11t17:33:29 товары 328 мяч 16 1723
1cfd733acdc38be"/>
```

Голубым цветом выделен артикул товара. Желтым цветом выделено содержимое двух атрибутов: *версия объекта* и *дата создания объекта*.

Искомая подстрока:

```
«16 2018»
```

Объект будет ошибочно выбран, т.к. взаимное расположение и значения атрибутов *версия объекта* и *дата создания объекта* образуют искомую подстроку.

Дополнительные сведения:

1) Поскольку поиск подстроки осуществляется в индексе *~Search~*, а итоговое XML-описание объектов берется из другого индекса, то оно может не содержать атрибутов с искомой подстрокой.

2) Поиск подстроки осуществляется без учета регистра символов.

3) Дополнительный параметр может содержать несколько подстрок, разделенных пробелами. В этом случае правила отбора объектов определяется значением параметра *allwords*.

4) Для составления сложных условий отбора, в дополнительном параметре можно использовать знаки логических операций, которые ставятся между искомыми подстроками. Искомые подстроки должны отделяться пробелами от знаков логических операций. Операторы **или**, **or**, **|** (вертикальная черта) соответствуют логической операции «ИЛИ». Операторы **и**, **and**, **&** соответствуют логической операции «И». Если искомые подстроки разделены только пробелами, то логические операции между ними определяются параметром *allwords*. По правилам логики операция «И» имеет более высокий приоритет (выполняется раньше), чем операция «ИЛИ», чтобы изменить порядок выполнения логических операций необходимо использовать круглые скобки.

5) Если отсутствуют объекты, удовлетворяющие условию поиска, то команда возвращает пустую строку.

6) Если искомая подстрока содержит символы пробелов, то ее необходимо заключать в двойные или одинарные кавычки.

Примечание – В текущей версии платформы ODANT, если условие поиска содержит несколько подстрок разделенных пробелами, причем одна из подстрок заключена в кавычки, то команда может выдавать сообщение об ошибке. Условие возникновения ошибки следующее:

– условие поиска содержит несколько подстрок разделенных пробелами, причем одна из подстрок заключена в кавычки и находится во второй или последующих позициях.

В текущей версии ODANT, чтобы команда всегда выполнялась корректно, необходимо выполнить одно из правил:

– подстрока в кавычках должна стоять первой в условии;

– подстрока в кавычках должна быть заключена в круглые скобки;

– между подстроками должны быть явно указаны соответствующие логические операции.

Пример условия поиска, которое вызовет ошибку при выполнении команды. Ищем в каталоге все мячи и товар с артикулом «16 2018»:

`мяч «16 2018»` (условие вызовет ошибку)

Возможны три способа изменения записи условия, устраняющие ошибку:

1) `мяч` **`or`** `«16 2018»` (явно указали логический оператор **or** между подстроками)
2) `мяч («16 2018»)` (подстроку в кавычках заключили в круглые скобки)
3) `«16 2018» мяч` (подстроку в кавычках поставили на первое место)

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса с указанием наименования индекса, который необходимо использовать.
loadmask	строка	Маска имен пакетов объектов, в которых осуществляется поиск (см. подробное описание параметра во Команды работы с индексами).
allwords	bool	Используется, если дополнительный параметр содержит подстроки не связанные логическими операторами, а только пробелами. Значение: *true* – выбираются только объекты, атрибуты которых содержат все подстроки, пробел соответствует операции "И"; *false* (по умолчанию) – выбираются все объекты, атрибуты которых содержат хотя бы одну из этих подстрок, пробел соответствует операции "ИЛИ".

Дополнительный параметр

Условие поиска, состоящее из искомых подстрок, соединенных пробелами и/или знаками логических операций.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – XML	**Тип** – определяется параметром *format*

Набор отдельных тегов с XML-описанием объектов, удовлетворяющих условию поиска.	Структура с описанием объектов, удовлетворяющих условию поиска:
	`{`
	`"$TagName":[`
	`{`
	`"attribute_1":"value_1_1",`
	`"attribute_2":"value_1_2",`
	`--//--//--`
	`"attribute_N":"value_1_N",`
	`},`
	`--//--//--`
	`{`
	`"attribute_1":"value_M_1",`
	`"attribute_2":"value_M_2",`
	`--//--//--`
	`"attribute_N":"value_M_N",`
	`}`
	`]`
	`}`
	где `$TagName` — имя тегов с описанием объектов в индексе, перед именем тега стоит символ доллара;
	`attribute_x` — имена атрибутов в индексе;

	`value_x_y` — значения соответствующих атрибутов.

Пример команды в Windows-клиенте

Команда:

```
ODANT ->
search_index?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D3E8669A067B9E/I:MY_IND
EX&loadmask=1965 (0 мс)
```

Дополнительный параметр:

```
Абрамов | Петров
```

Результат выполнения:

```
<O oid="1D3F35A588D85D1" Фамилия="Абрамов" Имя="Петр" Отчество="Юрьевич"/>
<O oid="1D3E935076E4CB8" Фамилия="Петров" Имя="Петр" Отчество="Петрович"/>
```

Примечание – Объекты класса *Клиенты* разбиты по пакетам. Объекты объединены в пакеты по году рождения клиента. Параметр *loadmask=1965* указывает, что поиск осуществляется среди лиц, рожденных в 1965 году. Выбираются только клиенты с фамилиями *Абрамов* или *Петров*.

Пример команды в программе Postman

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0?me
thod=search_index&id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:M
Y_INDEX&allwords=true
```

Дополнительный параметр:

```
Иван Иванович
```

Результат выполнения:

```
{
    "$O": [
        {
            "oid": "1D3F35D577229A0",
            "Фамилия": "Иванов",
            "имя": "Иван",
            "Отчество": "Иванович"
        },
        {
            "oid": "1D3F36C33ADC4C0",
            "Фамилия": "Кузнецов",
            "имя": "Иван",
            "Отчество": "Иванович"
        }
    ]
}
```

Примечание – Объекты класса *Клиенты* разбиты по пакетам. Параметр *loadmask* не указан, поэтому поиск осуществляется во всех пакетах. Параметр *allwords=true* указывает, что объект должен содержать обе подстроки, т.е. клиент должен быть обязательно Иваном Ивановичем.

3.4 update_index

Выражение

```
XML update_index(string id, string loadmask)
```

Rest API

```
SourceWebID?method=update_index&id=ClassFullID/I:IndexName
```

Описание

Команда обновляет указанный индекс класса. Если индекс не существует, то он создается. Индекс обновляется только в том случае, если с момента последнего обновления индекса изменился XQuery-запрос или объекты класса. Если не произошло изменений XQuery-запроса или объектов класса, то команда не изменяет индекс. Команда учитывает разбиение объектов на пакеты и не обновляет индексы для пакетов, объекты в которых не изменились.

Если класс имеет дочерние классы, расположенные в дочерних доменах, то команда, примененная к родительскому классу, включает в индекс родительского класса индексы из дочерних классов.

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса с указанием наименования индекса, который необходимо обновить.
loadmask	строка	Маска имен пакетов объектов, индексы которых могут быть обновлены в случае необходимости (см. подробное описание параметра во Команды работы с индексами).

Дополнительный параметр

Не используется.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – XML	~~**Тип** – определяется параметром *format*~~
Набор отдельных XML-тегов с информацией об обновленных и созданных файлах, содержащих индекс класса. Формат тега: `<F n='путь' t='время'>` где `путь` – имя файла индекса с относительным путем. Путь указывается относительно корневого	Информация об обновленных и созданных файлах, содержащих индекс класса. В текущей версии платформы ODANT, формат возвращаемого значения совпадает в Windows- и

каталога индекса, сам корневой каталог в пути не указывается (см. пример ниже); `время` – время создания файла индекса. Время выражается в количестве тактов по 100 нс прошедших с 1600 года.	Web-клиентах. В будущих версиях ODANT команда будет возвращать значение в формате, заданном в параметре *format*.

Пример команды в Windows-клиенте

Команда:

```
ODANT ->
update_index?id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:MY_IND
EX&loadmask=196* (8 мс)
```

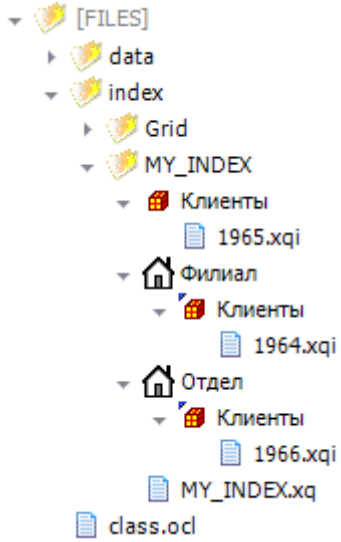
Дополнительный параметр:

```
Не используется.
```

Результат выполнения:

```
<F n='d.1D3F81E7505AD50\1D3F35D05DF2AC0\1964.xqi' t='131721632574922001'/><F
n='1D3F35D05DF2AC0\1965.xqi' t='131721632574922002'/><F
n='d.1D3F81EC918FE10\1D3F35D05DF2AC0\1966.xqi' t='131721632574922000'/>
```

Примечание – Объекты класса *Клиенты* разбиты по пакетам. Объекты объединены в пакеты по году рождения клиента. Класс *Клиенты* имеет одноименные дочерние классы в дочерних доменах-базах *Филиал* и *Отдел*. Домены-базы имеют идентификаторы 1D3F81E7505AD50 и 1D3F81EC918FE10. Параметр *loadmask=196** указывает, что индексы должны быть обновлены только для клиентов, рожденных в 60-е годы. Атрибут «n» в отчете содержит путь к файлу с индексом. На рисунке показано расположение созданных файлов с индексами относительно корневой папки класса.



Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0?me
thod=update_index&id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:M
Y_INDEX&loadmask=196*
```

Дополнительный параметр:

```
Не используется.
```

Результат выполнения:

1) создаются и обновляются требуемые файлы индексов;
2) в текущей версии платформы ODANT формат возвращаемого значения не может быть обработан Web-браузером, поэтому отображается сообщение об ошибке:

Пример команды в программе Postman

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0?me
thod=update_index&id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:M
Y_INDEX&loadmask=196*
```

Дополнительный параметр:

```
Не используется.
```

Результат выполнения:

```
<F n='d.1D3F81E7505AD50\1D3F35D05DF2AC0\1964.xqi' t='131721632574922001'/>
<F n='1D3F35D05DF2AC0\1965.xqi' t='131721632574922002'/>
<F n='d.1D3F81EC918FE10\1D3F35D05DF2AC0\1966.xqi' t='131721632574922000'/>
```

Примечание – Объекты класса *Клиенты* разбиты по пакетам. Объекты объединены в пакеты по году рождения клиента. Класс *Клиенты* имеет одноименные дочерние классы в дочерних доменах-базах *Филиал* и *Отдел*. Домены-базы имеют идентификаторы 1D3F81E7505AD50 и 1D3F81EC918FE10. Параметр *loadmask=196** указывает, что индексы должны быть обновлены только для клиентов, рожденных в 60-е годы. Атрибут «n» в отчете содержит путь к файлу с индексом.

3.5 xquery_index

Выражение

```
XML xquery_index(string id, string loadmask)
```

Rest API

```
SourceWebID?method=xquery_index&id=ClassFullID/I:IndexName
```

Описание

Команда выдает индекс с указанным именем (при отсутствии XQuery-запроса) или результат выполнения XQuery-запроса над этим индексом. XQuery-запрос предается в дополнительном параметре. В случае необходимости при выполнении команды

происходит обновление индекса в файловой системе базы данных (см. команду *update_index*).

В процессе работы команда осуществляет несколько операций:

1) объединяет все требуемые пакеты индекса в один XML-документ. Этот документ имеет несколько корневых тегов, по числу обработанных пакетов с индексами. Фактически происходит копирование в один XML-документ содержимого всех пакетов индексов, соответствующих маске указанной в параметре *loadmask*;
2) применяет к созданному XML-документу XQuery-запрос, если он предан в дополнительном параметре;
3) если команда выполняется в Windows-клиенте, то выдает полученный результат и завершает работу;
4) если команда выполняется в Web-клиенте и должна возвращать значение в JSON-формате, то команда дополнительно преобразует полученный XML-документ или результат его обработки XQuery-запросом в JSON-формат, и выдает его в качестве результата.

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса с указанием наименования индекса, который необходимо получить.
loadmask	строка	Маска имен файлов с пакетами индексов, которые обрабатываются командой (см. подробное описание параметра во Команды работы с индексами).

Дополнительный параметр

XQuery-запрос к индексу.

Примечания

1) Команда, выполняемая в Windows-клиенте, не подвергает результат XQuery-запроса никакому преобразованию и выдает его «как есть». Поэтому для нее допустим любой синтаксически правильный XQuery-запрос.
2) Команда, выполняемая в Web-клиенте, пытается преобразовать результат XQuery-запроса в JSON-формат, если это требуется форматом возвращаемого значения. В случае невозможности такого преобразования команда возвращает пустую строку. Например, если XQuery-запрос содержит оператор `document{}` для создания стандартного заголовка XML-документа `<?xml version="1.0"?>`, то команда возвратит пустую строку, т.к. не сможет включить заголовок в JSON-структуру. Будьте внимательны при составлении XQuery-запроса для команды в Web-клиенте.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – XML	**Тип** – определяется параметром *format*
1) При отсутствии XQuery-запроса, возвращает XML-документ, объединяющий все требуемые пакеты индекса. Этот документ имеет стандартный заголовок `<?xml version="1.0"?>`. 2) При наличии XQuery-запроса, возвращает результат выполнения этого запроса над документом, полученным в пункте 1.	1) При отсутствии XQuery-запроса, возвращает сам индекс. 2) При наличии XQuery-запроса, возвращает результат выполнения этого запроса над индексом.

Пример №1 команды в Windows-клиенте

Команда:

```
ODANT ->
xquery_index?id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:MY_IND
EX&loadmask=196* (9 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
<?xml version="1.0"?>
    <x cid="1D3F35D05DF2AC0" bid="1D3F81EC918FE10" pack="1964">
                <O oid="1D3FE3F8B285907" Фамилия="Петров" Имя="Петр"
Отчество="Петрович"/>
    </x>
    <x cid="1D3F35D05DF2AC0" bid="1D1186B9A86E3D6" pack="1965">
        <O oid="1D3FE3F488DB83D" Фамилия="Абрамов" Имя="Петр" Отчество="Юрьевич"/>
        <O oid="1D3FE3F4EB58732" Фамилия="Петров" Имя="Петр" Отчество="Петрович"/>
    </x>
    <x cid="1D3F35D05DF2AC0" bid="1D3F81E7505AD50" pack="1965">
        <O oid="1D3FE3F0307A75C" Фамилия="Долин" Имя="Сергей" Отчество="Иванович"/>
    </x>
    <x cid="1D3F35D05DF2AC0" bid="1D3F81E7505AD50" pack="1966">
        <O oid="1D3FE3F166F7F38" Фамилия="Петров" Имя="Петр" Отчество="Петрович"/>
    </x>
```

Пример №2 команды в Windows-клиенте

Команда:

```
ODANT ->
xquery_index?id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:MY_
INDEX&loadmask=196* (15 мс)
```

Дополнительный параметр:

```
document{
    element customers {
        attribute cid {
            x[1]/@cid
        },
        for $pk in /x
```

```
        for $person in $pk/O
        return
            element customer {
                attribute last_name {
                    $person/@Фамилия
                },
                attribute year {
                    $pk/@pack
                }
            }
    }
}
```

Результат выполнения:

```
<?xml version="1.0"?>
    <customers cid="1D3F35D05DF2AC0">
        <customer last_name="Петров" year="1964"/>
        <customer last_name="Абрамов" year="1965"/>
        <customer last_name="Петров" year="1965"/>
        <customer last_name="Долин" year="1965"/>
        <customer last_name="Петров" year="1966"/>
    </customers>
```

Примечание - Команда во втором примере отличается от команды в первом примере наличием XQuery-запроса, переданного в дополнительном параметре. Поэтому результат выполнения команды во втором примере соответствует выполнению XQuery-запроса над результатами выполнения команды в первом примере.

Пример №3 команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0?method=xquery_index&id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:MY_INDEX&loadmask=196*
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:


```
{"$x":[{"cid":"1D3F35D05DF2AC0","bid":"1D3F81EC918FE10","pack":"1964","$O":[{"oid":"1D3FE3F8B285907","Фамилия":"Петров","Имя":"Петр","Отчество":"Петрович"}]},{"cid":"1D3F35D05DF2AC0","bid":"1D1186B9A86E3D6","pack":"1965","$O":[{"oid":"1D3FE3F488DB83D","Фамилия":"Абрамов","Имя":"Петр","Отчество":"Юрьевич"},{"oid":"1D3FE3F4EB58732","Фамилия":"Петров","Имя":"Петр","Отчество":"Петрович"}]},{"cid":"1D3F35D05DF2AC0","bid":"1D3F81E7505AD50","pack":"1965","$O":[{"oid":"1D3FE3F0307A75C","Фамилия":"Долин","Имя":"Сергей","Отчество":"Иванович"}]},{"cid":"1D3F35D05DF2AC0","bid":"1D3F81E7505AD50","pack":"1966","$O":[{"oid":"1D3FE3F166F7F38","Фамилия":"Петров","Имя":"Петр","Отчество":"Петрович"}]}]}
```


Пример №4 команды в программе Postman

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0?method=xquery_index&id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:MY_INDEX&loadmask=196*
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:


```
{
    "$x": [
        {
            "cid": "1D3F35D05DF2AC0",
            "bid": "1D3F81EC918FE10",
            "pack": "1964",
            "$O": [
                {
                    "oid": "1D3FE3F8B285907",
                    "Фамилия": "Петров",
                    "Имя": "Петр",
                    "Отчество": "Петрович"
                }
            ]
        },
        {
            "cid": "1D3F35D05DF2AC0",
            "bid": "1D1186B9A86E3D6",
            "pack": "1965",
            "$O": [
                {
                    "oid": "1D3FE3F488DB83D",
                    "Фамилия": "Абрамов",
                    "Имя": "Петр",
                    "Отчество": "Юрьевич"
                },
                {
                    "oid": "1D3FE3F4EB58732",
                    "Фамилия": "Петров",
                    "Имя": "Петр",
                    "Отчество": "Петрович"
                }
            ]
        },
        {
            "cid": "1D3F35D05DF2AC0",
            "bid": "1D3F81E7505AD50",
            "pack": "1965",
            "$O": [
                {
                    "oid": "1D3FE3F0307A75C",
                    "Фамилия": "Долин",
                    "Имя": "Сергей",
                    "Отчество": "Иванович"
                }
            ]
        },
        {
            "cid": "1D3F35D05DF2AC0",
            "bid": "1D3F81E7505AD50",
            "pack": "1966",
            "$O": [
                {
                    "oid": "1D3FE3F166F7F38",
                    "Фамилия": "Петров",
                    "Имя": "Петр",
                    "Отчество": "Петрович"
                }
            ]
        }
```

```
    ]
}
```

Пример №5 команды в программе Postman

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/~/client/pages/navigator/index.html?method=xquery_index&id=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/I:MY_INDEX&loadmask=196*
```

Дополнительный параметр:

```
element customers {
    attribute cid {
        x[1]/@cid
    },
    for $pk in /x
    for $person in $pk/O
    return
        element customer {
            attribute last_name {
                $person/@Фамилия
            },
            attribute year {
                $pk/@pack
            }
        }
}
```

Результат выполнения:

```
{
    "$customers": [
        {
            "cid": "1D3F35D05DF2AC0",
            "$customer": [
                {
                    "last_name": "Петров",
                    "year": "1964"
                },
                {
                    "last_name": "Абрамов",
                    "year": "1965"
                },
                {
                    "last_name": "Петров",
                    "year": "1965"
                },
                {
                    "last_name": "Долин",
                    "year": "1965"
                },
                {
                    "last_name": "Петров",
                    "year": "1966"
                }
            ]
        }
    ]
}
```

4 Команды работы с классами

4.1 change_class_parent

Выражение

```
bool change_class_parent( string newparent, string id )
```

Rest API

```
SourceWebID?method=change_class_parent&newparent=DestinationClassID
```

Описание

Команда переподчиняет указанный класс или домен новому родительскому классу или домену. Фактически команда копирует указанный класс или домен в новый родительский класс или домен, а затем удаляет указанный класс из прежнего родительского класса (домена). При этом в файловой структуре базы данных происходят следующие изменения:

1) подкаталог с классом (доменом) копируется в каталог нового родительского класса (домена);

2) подкаталог с классом (доменом) не удаляется из каталога исходного родительского класса (домена), а перемещаются в подкаталог «.removed», расположенный в каталоге исходного родительского класса (домена).

Раздел не может содержать домены с одинаковыми идентификаторами, а домен не может содержать классы с одинаковыми идентификаторами, поэтому команда возвращает сообщение об ошибке, если при ее выполнении может возникнуть нарушение этого правила.

Команда выполняет переподчинение доменов и классов только в пределах одного хоста, перемещение доменов и классов между разными хостами невозможно.

Примечания

1) Команда не проверяет соответствие типов разделов и типов, перемещаемых в них доменов. Например, команда может переместить домен-разработчика из раздела *Разработка* в раздел *Общий*.

2) В текущей версии платформы ODANT при перемещении класса (домена) в домен-рабочее место, команда сообщает об успешном выполнении операции, перемещаемый класс (домен) удаляется из структуры предыдущего родителя, но в домен-рабочее место перемещаемый класс (домен) не копируется.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
newparent	строка	Полный идентификатор нового родительского класса или домена.
id	строка	Полный идентификатор перемещаемого класса или домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – логический	**Тип** – определяется параметром *format*

`true` – если команда выполнена успешно
`сообщение об ошибке` – если команда не может быть выполнена.
	Если команда выполнена успешно:
	`{`
	`"result":"true"`
	`}`
	
	Если команда не может быть выполнена:
	`сообщение об ошибке`

Пример команды в Windows-клиенте

Команда:

```
moved_class [Мой компьютер/MyTes.] ->
change_class_parent?newparent=H:1D0EB94C5ACD42E/D:WORK/D:1D39B433C737C8C (0 мс)
```

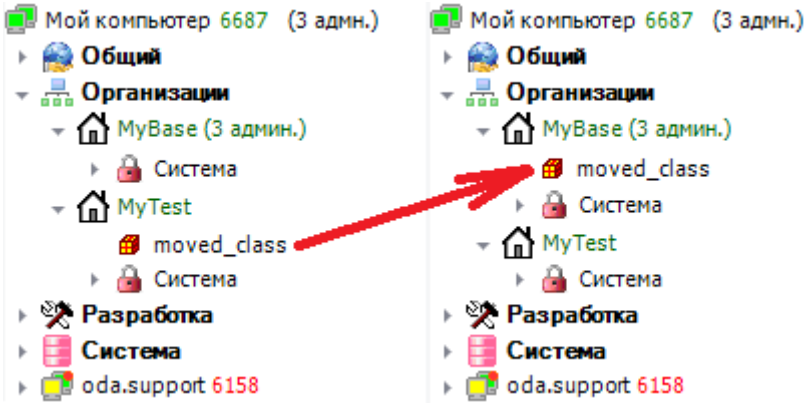
Дополнительный параметр:

```
Не используется.
```

Результат выполнения:

```
true
```

Примечание – Команда переместила класс *moved_class* из домена-базы *MyTest* домен-базу *MyBase*.



Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D3F81EC918FE10/C:1D40327177DE710?me
thod=change_class_parent&newparent=H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6
```

Дополнительный параметр:

```
Не используется.
```

Результат выполнения:

```
{"result":"true"}
```

4.2 create_class

Выражение

```
string create_class(string name, string id)
```

Rest API

```
SourceWebID?method=create_class&name=NewClassName
```

Описание

Команда создает дочерний класс в указанном родительском классе или домене. Команда позволяет создать:

— «пустой» класс (класс без внутренней структуры). Для этого достаточно в параметре команды *name* указать имя создаваемого класса, а дополнительный параметр не используется. Параметр *name* можно опустить. В этом случае созданный класс будет иметь имя по умолчанию «Класс » (обратите внимание, что указанное имя содержит 6 символов, т.к. последний символ является пробелом);

— класс на основании XML-шаблона. Шаблон передается команде через дополнительный параметр. В этом случае параметр *name* игнорируется, а имя класса задается в шаблоне. Тэг <*CLASS*> в шаблоне обязательно должен содержать атрибуты *ClassId* и *Name*, при их отсутствии выдается сообщение об ошибке. Если класс с указанным идентификатором уже существует в родительском элементе, заданном в параметре *id*, то его XML-описание будет замещено новым шаблоном. Прежнее XML-описание класса не удаляется, а помещается в архив класса и может быть

восстановлено. Замена XML-описания класса не влияет на существующие объекты класса.

При создании класса из XML-шаблона команда автоматически добавляет в тег <*CLASS*> атрибуты *Date* и *Update*. Если указанные атрибуты присутствуют в XML-шаблоне, то в XML-описании созданного класса их значения будут заменены на актуальные значения.

При создании класса без использования XML-шаблона, но с использованием параметра *name*, команда создает следующее XML-описание класса:

```
<?xml version="1.0"?>

      <CLASS Author="1CFD733ACDC38BE" ClassId="1D407CD75ACC4F0" Date="2018-06-
19T16:00:12" Name="CreatedClass" Label="CreatedClass" Parent="1D1186B9A86E3D6"
Update="2018-06-19T16:00:12"/>
```

Примечание – Атрибуты *Name* и *Label* всегда имеют значение, переданное команде в параметре *name*. Конкретные значения остальных атрибутов зависят от контекста, в котором выполнена команда.

При создании класса без использования XML-шаблона и без использования параметра *name*, команда создает следующее XML-описание класса:

```
<?xml version="1.0"?>

      <CLASS Author="1CFD733ACDC38BE" ClassId="1D407CF71AF5640" Date="2018-06-
19T16:14:24" Parent="1D1186B9A86E3D6" Name="Класс " Update="2018-06-19T16:14:24"/>
```

Примечание – Атрибут *Name* всегда имеет значение "Класс ". Конкретные значения остальных атрибутов зависят от контекста, в котором выполнена команда.

Примечания

1) Команда не проверяет тип домена или раздела, в котором создается класс. Например, команда может создать класс в корне раздела *Организации*, хотя визуальная среда разработки не позволяет этого делать.

2) В текущей версии платформы ODANT при создании класса в домене-рабочее место команда успешно создает класс в файловой структуре этого домена, но в проводнике созданный класс не отображается.

3) В текущей версии платформы ODANT, если в команде опущен параметр *name* и отсутствует шаблон класса в дополнительном параметре, то команда не создает атрибут *Label* в теге <*CLASS*> в XML-описании класса.

4) Будьте внимательны при выборе идентификатора создаваемого класса в XML-шаблоне. Это связано с тем, что домен не может содержать классы с одинаковыми идентификаторами (см. документ о принципах адресации в ODANT). Если домен, в котором создается класс, уже содержит класс с тем же идентификатором, то команда проигнорирует значение параметра id и заменит XML-описание уже существующего класса.

Например, рассмотрим последовательное выполнение двух команд.

Команда №1, по нашим задумкам, должна создать дочерний класс «Адрес» в родительском классе с идентификатором 222222222222222:

```
      create_class?id=H:1CFD733ACDC38BE/D:WORK/D:111111111111111/C:222222222222222
дополнительный параметр:
      <?xml version="1.0"?>
            <CLASS ClassId="333333333333333" Name="Адрес">
                  <SF/>
                  <METADATA>
                        <ATTR Name="Город" Label="Город" List="False"/>
                  </METADATA>
            </CLASS>
```

Команда №2, по нашим задумкам, должна создать дочерний класс «Клиент» в родительском классе с идентификатором 444444444444444:

```
      create_class?id=H:1CFD733ACDC38BE/D:WORK/D:111111111111111/C:444444444444444
дополнительный параметр:
      <?xml version="1.0"?>
            <CLASS ClassId="333333333333333" Name="Клиент">
                  <SF/>
                  <METADATA>
                        <ATTR Name="Фамилия" Label="Фамилия" List="False"/>
                  </METADATA>
            </CLASS>
```

Но идентификаторы обоих создаваемых классов совпадают. Поэтому первая команда создаст требуемый класс «Адрес», как и было задумано. Вторая команда сообщит, что выполнилась успешно, но не создаст дочерний класс в ожидаемом родительском классе с идентификатором 444444444444444, как мы рассчитывали, а заменит XML-описание класса созданного первой командой.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
name	строка	Имя создаваемого класса.
id	строка	Полный идентификатор класса или домена, в котором создается класс.

Дополнительный параметр

XML-шаблон создаваемого класса.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – строка	**Тип** – определяется параметром *format*

Краткий идентификатор созданного класса.	Структура с описанием атрибутов созданного класса:
	`{"$C":[`
	`{`
	`"i":"ClassID",`

	<pre>"n":"ClassName",
	"a":"AccessLevel"
	}
	]}</pre>где $C — фиксированное имя группирующего ключа;
	i, n, a — фиксированные ключи *identifier*, *name* и *access level*;
	ClassID — краткий идентификатор созданного класса;
	ClassName — имя созданного класса;
	AccessLevel — уровень доступа, с которым была выполнена команда.
	Если создаваемый класс является абстрактным, то возвращаемая структура дополнительно содержит ключ ab со значением уровня абстракции (см. команду *get_class_abstract_level*).
	Также JSON-структура может содержать другие ключи, соответствующие атрибутам в теге <*CLASS*> в XML-описании созданного класса (например, ключ «l», который соответствует атрибуту *Label*).

Пример №1 команды в Windows-клиенте

Команда:

```
TestClass [Мой компьютер/ООО Солныш.] -> create_class?name=CreatedClass (684 мс)
```

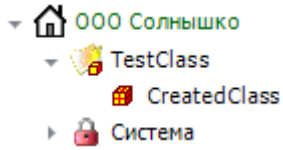
Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
1D16FA9661A42D3
```

Примечание – Команда создает «пустой» дочерний класс с именем *CreatedClass* в родительском классе *TestClass*. Созданному классу автоматически присвоен идентификатор *1D16FA9661A42D3*.



Пример №2 команды в Windows-клиенте

Команда:

```
TestClass [Мой компьютер/ООО Солныш.] -> create_class (15 мс)
```

Дополнительный параметр:

```
<?xml version="1.0"?>
    <CLASS ClassId="1D0F54787BAA8B4" Name="CreatedClass">
        <SF/>
        <METADATA>
            <ATTR Name="Город" Label="Город" List="False"/>
            <ATTR Name="Улиц" Label="Улица" List="False"/>
            <ATTR Name="Дом" Label="Дом" List="False"/>
            <ATTR Name="Квар" Label="Квартира" List="False"/>
        </METADATA>
```

```
    </CLASS>
```

Результат выполнения:

```
1D0F54787BAA8B4
```

Примечание – Команда создает дочерний класс с именем *CreatedClass* и идентификатором *1D0F54787BAA8B4* в родительском классе *TestClass*. Созданный класс будет содержать поля: «Город», «Улица», «Дом», «Квартира».

Пример №3 команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D406FC1AABB970?method=create_class&name=CreatedClass
```

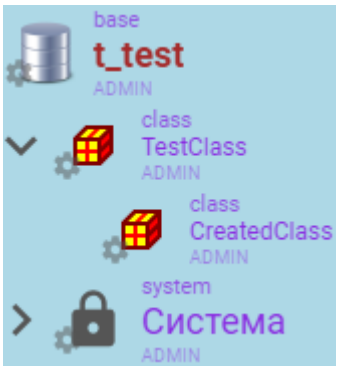
Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
{"$C":[{"a":"6","i":"1D406FD9DAD7470","n":"CreatedClass","ab":"1"}]}
```

Примечание – Команда создает «пустой» дочерний класс с именем *CreatedClass* в родительском классе *TestClass*. Идентификатор родительского класса *1D406FC1AABB970*. Созданному классу автоматически присвоен идентификатор *1D406FD9DAD7470*. Класс является абстрактным и имеет уровень абстракции 1.



Пример №4 команды в программе Postman

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D406FC1AABB970?method=create_class
```

Дополнительный параметр:

```
<?xml version="1.0"?>
    <CLASS ClassId="1D0F54787BAA8B4" Name="CreatedClass">
        <SF/>
        <METADATA>
            <ATTR Name="Город" Label="Город" List="False"/>
            <ATTR Name="Улиц" Label="Улица" List="False"/>
            <ATTR Name="Дом" Label="Дом" List="False"/>
            <ATTR Name="Квар" Label="Квартира" List="False"/>
        </METADATA>
    </CLASS>
```

Результат выполнения:

```
{
    "$C": [
        {
            "a": "6",
            "i": "1D0F54787BAA8B4",
            "n": "CreatedClass"
        }
    ]
}
```

Примечание – Команда создает дочерний класс с именем *CreatedClass* и идентификатором *1D0F54787BAA8B4* в родительском классе *TestClass*. Идентификатор родительского класса *1D406FC1AABB970*. Созданный класс будет содержать поля: «Город», «Улица», «Дом», «Квартира».

4.3 delete_class

Выражение

```
bool delete_class(string id)
```

Rest API

```
SourceWebID?method=delete_class
```

Описание

Команда удаляет указанный класс из структуры базы данных.

Примечания:

1) Подкаталог с классом не удаляется из каталога родительского класса, а перемещается в подкаталог «.removed», расположенный в каталоге родительского класса;

2) Данной командой нельзя удалить ссылку на класс, расположенную в домене-рабочем месте. Если в параметре *id* указать полный идентификатор ссылки на класс, то вместо ссылки будет удален сам класс, а неработающая ссылка останется в домене-рабочем месте.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор удаляемого класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Windows-клиент	Web-клиент
Тип – логический	**Тип** – определяется параметром *format*

`true` – если операция выполнена успешно;
`сообщение об ошибке` – если операция не выполнена.
	Если операция выполнена успешно:
	`{`
	`"result":"true"`
	`}`
	
	Если операция не выполнена:
	`сообщение об ошибке`

Пример команды в Windows-клиенте

Команда:

```
TestClass [Мой компьютер/ООО Солныш.] -> delete_class (15 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
true
```

Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D2BCF5BFC23940method=delete_class
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
{"result":"true"}
```

4.4 get_class

Выражение

```
XML get_class(string id)
```

Rest API

```
SourceWebID?method=get_class
```

Описание

Команда возвращает XML-описание указанного класса, домена или раздела (при отсутствии XQuery-запроса) или результат выполнения XQuery-запроса над этим XML-описанием. XQuery-запрос предается в дополнительном параметре.

Примечание – В текущей версии платформы ODANT команда возвращает XML-описания классов, доменов и разделов, к которым пользователь не имеет доступа и

которые, соответственно, не отображаются в проводнике, но известны их полные идентификаторы.

Необходимые права

Read

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, домена или раздела.

Дополнительный параметр

XQuery-запрос к. XML-описанию класса.

Примечания

1) Команда, выполняемая в Windows-клиенте, не подвергает результат XQuery-запроса никакому преобразованию и выдает его «как есть». Поэтому для нее допустим любой синтаксически правильный XQuery-запрос.
2) Команда, выполняемая в Web-клиенте, пытается преобразовать результат XQuery-запроса в JSON-формат, если это требуется форматом возвращаемого значения. В случае невозможности такого преобразования команда возвращает пустую строку. Например, если XQuery-запрос содержит оператор `document{}` для создания стандартного заголовка XML-документа `<?xml version="1.0"?>`, то команда возвратит пустую строку, т.к. не сможет включить заголовок в JSON-структуру. Будьте внимательны при составлении XQuery-запроса для команды в Web-клиенте.

Возвращаемое значение

Тип: XML

Описание: XML-описание класса, домена или раздела.

Windows-клиент	**Web-клиент**
Тип – XML	**Тип** – определяется параметром *format*

XML-описание класса, домена или раздела.	JSON-структура с описанием объектов, удовлетворяющих условию поиска:
	`{`
	`"$TagName":[`
	`{`
	`"attribute_1":"value_1_1",`
	`"attribute_2":"value_1_2",`
	`--//--//--`
	`"attribute_N":"value_1_N",`
	`},`
	`--//--//--`
	`{`

	"attribute_1":"value_M_1", "attribute_2":"value_M_2", --//--//-- "attribute_N":"value_M_N", }] } где $TagName — имя тегов с описанием объектов в индексе, перед именем тега стоит символ доллара; attribute_x — имена атрибутов в индексе; value_x_y — значения соответствующих атрибутов.

Пример команды в Windows-клиенте

Команда:

```
Клиенты [odant.org/t_tes.] -> get_class (2 мс)
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
<?xml version="1.0"?>
        <CLASS Author="1CFD733ACDC38BE" ClassId="1D3F35D05DF2AC0" Date="2018-05-
24T15:44:57" Name="Клиенты" Label="Клиенты" Parent="1D1186B9A86E3D6" Update="2018-
06-06T17:39:15" Pack="let $f := @Год let $Field := $f return $Field">
                <SF/>
                <METADATA>
                        <ATTR Name="Фамилия" Label="Фамилия" List="False"/>
                        <ATTR Name="Имя" Label="имя" List="False"/>
                        <ATTR Name="Отчество" Label="Отчество" List="False"/>
                        <ATTR Name="Год" Label="Год" List="False"/>
                </METADATA>
                <Pack id="261007664006865" Name="По полю: &amp;apos;@Год&amp;apos; [@Год]"
PackMode="Field" Field="@Год" Hide="False"/>
        </CLASS>
```

Пример команды в Web-браузере

Команда:

```
https://odant.org/api/H:1D03A3F3B5863BB/P:WORK/B:1D1186B9A86E3D6/C:1D3F35D05DF2AC0/~/
client/pages/navigator/index.html?method=get_class
```

Дополнительный параметр:

```
Не используется
```

Результат выполнения:

```
{
    "$CLASS": [
        {
            "Author": "1CFD733ACDC38BE",
            "ClassId": "1D3F35D05DF2AC0",
            "Date": "2018-05-24T15:44:57",
            "Name": "Клиенты",
            "Label": "Клиенты",
            "Parent": "1D1186B9A86E3D6",
            "Update": "2018-06-06T17:39:15",
```

```
            "Pack": "let $f := @Год let $Field := $f return $Field",
            "$SF": [
                {}
            ],
            "$METADATA": [
                {
                    "$ATTR": [
                        {
                            "Name": "Фамилия",
                            "Label": "Фамилия",
                            "List": "False"
                        },
                        {
                            "Name": "Имя",
                            "Label": "имя",
                            "List": "False"
                        },
                        {
                            "Name": "Отчество",
                            "Label": "Отчество",
                            "List": "False"
                        },
                        {
                            "Name": "Год",
                            "Label": "Год",
                            "List": "False"
                        }
                    ]
                }
            ],
            "$Pack": [
                {
                    "id": "261007664006865",
                    "Name": "По полю: '@Год' [@Год]",
                    "PackMode": "Field",
                    "Field": "@Год",
                    "Hide": "False"
                }
            ]
        }
    ]
}
```


```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_class
```

4.5 get_class_abstract_level

Выражение

```
number get_class_abstract_level(string id)
```

Rest API

```
SourceWebID?method=get_class_abstract_level
```

Описание

Команда возвращает код уровня абстракции класса:

2 - класс абстрактный;

1 - класс возможно абстрактный (отсутствуют поля в классе);

0 - класс не абстрактный.

Команда определяет уровень абстракции класса по значению атрибута *Abstract* тега *CLASS* в XML-описании класса и по наличию полей в классе.

Команда возвращает код 2, если в теге *CLASS* присутствует атрибут *Abstract* со значением *True*, наличие полей в классе не проверяется.

Команда возвращает код 1, если атрибут *Abstract* отсутствует в теге *CLASS* или имеет значение *False*, и поля в классе отсутствуют.

Команда возвращает код 0, если атрибут *Abstract* отсутствует в теге *CLASS* или имеет значение *False*, но в классе присутствует хотя бы одно поле.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: число

Описание: Код уровня абстракции класса.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш.] -> get_class_abstract_level (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
1
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_class_abstract_level
```

4.6 get_class_access

Выражение

```
number get_class_access(string id)
```

Rest API

```
SourceWebID?method=get_class_access
```

Описание

Команда возвращает цифровой код уровня доступа к классу текущего пользователя. Соответствие кодов уровням доступа следующее:

0 - нет доступа
1 - Preview
2 - R
3 - RW
4 - RWC
5 - RWCD
6 - Admin

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: число

Описание: Цифровой код уровня доступа к классу.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш./Прием заявок/Класс.] -> get_class_access (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
5
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E?method=get_class_access
```

4.7 get_class_attr

Выражение

```
string get_class_attr(string id, string attr)
```

Rest API

```
SourceWebID?method=get_class_attr?attr=AttributeName
```

Описание

Команда выдает значение атрибута из XML-описания указанного класса или домена.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена.
attr	строка	Имя атрибута тега <*CLASS*> в XML-описании класса (домена).

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Значение атрибута.

Пример команды в Windows-клиенте

Команда

```
Организации [Мой компьютер] -> get_class_attr?attr=ClassId (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
WORK
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D0EB94C5ACD42E/D:WORK?method=get_class_attr?attr=ClassId
```

4.8 get_class_objects_count

Выражение

```
number get_class_objects_count(string id)
```

Rest API

```
SourceWebID?method=get_class_objects_count
```

Описание

Команда возвращает количество объектов в классе.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: число

Описание: Количество объектов в классе.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш./Класс.] -> get_class_objects_count (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
15
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_class_objects_count
```

4.9 get_class_parent

Выражение

```
string get_class_parent(string id)
```

Rest API

```
SourceWebID?method=get_class_parent
```

Описание

Команда возвращает полный идентификатор класса или домена, являющегося базовым для класса или домена переданного в параметре *id*.

Необходимые права

R

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого класса или домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Полный идентификатор родительского класса или домена.

Пример команды в Windows-клиенте

Команда 1

```
Организации [Мой компьютер] -> get_class_parent (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
/H:000000000000000/D:WORK/C:WORK
```

Примечание – Базовый класс домена-раздела «Организации» локального хоста расположен на хосте «oda.support».

Команда 2

```
Архив отчётов [Мой компьютер/ООО Солныш./Отчёт.2.0] -> get_class_parent (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
/H:1CC2A7562264BB7/D:ROOT/D:1CDA7834ECABDD2/C:1D054B658982F6F
```

Примечание – Базовый класс модуля «Архив отчетов», входящего в структуру локального хоста, расположен на хосте «oda.developer».

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK?method=get_class_parent
```

4.10 is_class_has_bin

Выражение

```
bool is_class_has_bin(string id)
```

Rest API

```
SourceWebID?method=is_class_has_bin?id=FullClassID
```

Описание

Команда проверяет наличие dll файла у модуля класса.

Необходимые права

R

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание

true - модуль класса имеет DLL файл;

false - модуль класса не имеет DLL файл.

Пример команды в Windows-клиенте

Команда

```
ODANT ->
is_class_has_bin?id=H:000000000000000/D:ROOT/D:1CD85229B0F580E/C:142779122248358 (0
мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание - Команда подтверждает, что класс *Печатные шаблоны* на хосте *oda.support* имеет DLL файл.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=is class has bin?id=H:000000000000000/D:ROOT/D:1CD85229B0F580E/C:142779122248
358
```

4.11 is_class_has_modules

Выражение

```
bool is_class_has_modules(string id)
```

Rest API

```
SourceWebID?method=is_class_has_modules?id=FullClassID
```

Описание

Команда проверяет наличие папки *modules* с исходными текстами модуля класса.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true – папка *modules* существует;

false – папка *modules* не существует.

Пример команды в Windows-клиенте

Команда

```
ODANT ->
is_class_has_modules?id=H:000000000000000/D:ROOT/D:1CD85229B0F580E/C:142779122248358
(0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание – Команда подтверждает, что класс *Печатные шаблоны* на хосте *oda.support* имеет папку *modules*.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=is_class_has_modules?id=H:000000000000000/D:ROOT/D:1CD85229B0F580E/C:14277912
2248358
```

4.12 is_override_class

Выражение

```
bool is_override_class(string id)
```

Rest API

```
SourceWebID?method=is_override_class
```

Описание

Команда выполняет проверку, что класс переопределяет методы, наследуемые от базового класса. Команда возвращает значение *true*, если в XML-описании класса тег *<CLASS>* содержит атрибут *OverrideClass*, имеющий значение *True*.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true – класс переопределяет методы наследуемые от базового класса;

false – класс не переопределяет методы наследуемые от базового класса.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш.] -> is_override_class (15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=is_override_class
```

4.13 save_class

Выражение

```
bool save_class(number tc, number lc, string id)
```

Rest API

```
SourceWebID?method=save_class
```

Описание

Команда сохраняет XML-описание нового класса. XML-описание класса передается команде в дополнительном параметре. Команда может сохранить только уже существующий класс, новый класс данной командой создать нельзя. С помощью этой команды можно удалить из класса старые поля и добавить новые поля данных. При удалении из класса полей, данные хранящиеся в этих полях в уже существующих объектах не удаляются, но доступ к ним через визуальный интерфейс платформы становится невозможным.

Команда фактически заменяет в базе данных старый файл с XML-описанием класса вновь создаваемым файлом, который содержит текст, переданный в дополнительном параметре.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
tc	число	ID транзакции, во время сохранения класса.
lc	число	ID блокировки, во время сохранения класса.
id	строка	Полный идентификатор сохраняемого класса.

Дополнительный параметр

XML-описание сохраняемого класса.

Возвращаемое значение

Тип: bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш./Класс.] -> save_class (0 мс)
```

Дополнительный параметр

```
<?xml version="1.0"?>
    <CLASS Author="1D0EB94C5ACD42E" ClassId="1D16BD3F1E36EDC" Date="2016-02-
20T14:43:41" Name="TestClass" Label="TestClass" Parent="1D0F538491111F9"
Update="2016-02-20T14:43:41">
        <SF/>
        <METADATA>
            <ATTR Name="Город" Label="Город" List="False"/>
        </METADATA>
    </CLASS>
```

Результат выполнения

```
true
```

Примечание – Команда удаляет все поля в классе за исключением поля «Город».

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=save_class
```

5 Команды работы с файлами

5.1 check_file

Выражение

```
bool check_file(string id, string filename)
```

Rest API

```
SourceWebID?method=check_file?filename=CheckedFileName
```

Описание

Команда проверяет наличие заданного файла в файловой структуре указанного класса или домена.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, с указанием имени проверяемого файла. Вместе с именем файла допускается указывать относительный путь к файлу. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена).
filename	строка	Имя проверяемого файла. Вместе с именем файла допускается указывать относительный путь к файлу.

Примечание - Имя проверяемого файла можно передать команде двумя способами:

1) через параметр *filename*, при этом имя файла в параметре *id* указывать не надо;

2) указать имя файла в параметре *id*, при этом параметр *filename* использовать не надо.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: Bool

Описание:

true – файл существует;

false – файл отсутствует.

Пример команды в Windows-клиенте

Команда 1

```
TestClass [Мой компьютер/ООО Солныш./Класс.] -> check_file?filename=data\pack.oml
(15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Команда 2

```
ODANT ->
check_file?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3F1E36EDC/F:data\pac
k.oml (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание - В первом примере имя проверяемого файла передается в параметре *filename*. Во втором примере имя проверяемого файла передается в параметре *id*.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
/help.php?method=check_file?filename=data\pack.oml
```

5.2 check_folder

Выражение

```
bool check_folder(string id)
```

Rest API

```
SourceWebID?method=check_folder?id=FullClassID/F:CheckedFolderName
```

Описание

Команда проверяет наличие заданной папки в файловой структуре указанного класса или домена.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, с указанием

Наименование параметра	Тип	Описание
		имени проверяемой папки. Вместе с именем папки допускается указывать относительный путь к папке. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена).

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: Bool

Описание:

true - папка существует.

false - папка не существует.

Пример команды в Windows-клиенте

Команда

```
ODANT ->
check_folder?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3F1E36EDC/F:..\.re
moved (15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
/help.php?method=check_folder?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3
F1E36EDC/F:..\.removed
```

5.3 copy_file

Выражение

```
bool copy_file(string id, string newfilename, string oldfilename, string tc)
```

Rest API

```
SourceWebID?method=move_file?oldfilename=SourceWebID&newfilename=DestinationWebID
```

Описание

Команда копирует существующий в базе данных файл в новое место. Если копируемый файл уже существует в папке назначения, то он будет перезаписан. Если файл в папке назначения имеет атрибут ReadOnly, то команда вернет сообщение об

ошибке и выполнена не будет. Если папка назначения, указанная в команде, не существует, то команда выполнена не будет.

Имена создаваемого и копируемого файлов указываются с относительными путями к ним. Если в качестве пункта назначения указан домен, то относительный путь начинается в каталоге *domain/class/* этого домена. Если в качестве пункта назначения указан класс, то относительный путь начинается в каталоге *class* этого класса.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, в файловой структуре которого находится копируемый файл.
oldfilename	строка	Имя копируемого файла с указанием относительного пути к нему.
newfilename	строка	Имя создаваемой копии файла с указанием относительного пути к месту ее расположения.
tc	строка	ID транзакции во время копирования файла.

Примечание - Использовать подстановочные символы "*" и "?" в именах исходного файла и файла-копии не допускается.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: Bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
TestClass01 [Мой компьютер/ООО Солныш.] ->
copy_file?oldfilename=class.ocl&newfilename=..\..\TestClass02\class\class.ocl (15
мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание – Команда копирует файл с XML-описанием класса *TestClass01* в файловую структуру соседнего класса *TestClass02*, заменяя его XML-описание. После выполнения команды классы *TestClass01* и *TestClass02* будут иметь одинаковую структуру и одинаковый идентификатор.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
/help.php?method=move_file?oldfilename=http://www.odant.org/api/H:1D03A3F3B5863BB/D:W
ORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E/help.htm&newfilename=http://www.odant.org/api
/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E/index.html
```

5.4 delete_bymask

Выражение

```
bool delete_bymask(string id, string filename, string tc, bool onlyfile)
```

Rest API

```
SourceWebID?method=delete_bymask?filename=FileMask[&onlyfile=true|false]
```

Описание

Команда удаляет файлы и папки из файловой структуры базы данных по заданной маске.

Примечание – В зависимости от значения параметра onlyfile, команда удаляет или только подкаталоги, расположенные в заданной директории, или только файлы. Т.е. чтобы удалить все файлы и все подкаталоги, соответствующие заданной маске, необходимо выполнить команду два раза с разными значениями параметра onlyfile.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, из файловой структуры которого удаляются файлы или папки.
filename	строка	Маска имен удаляемых файлов или папок. Здесь же можно указывать относительный путь к директории, на которую распространяется команда. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена).
tc	строка	ID транзакции во время удаления.
onlyfile	bool	Значение: true – удаляются только файлы; false (по умолчанию) – удаляются только подкаталоги.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: Bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда 1

```
Класс01 [Мой компьютер/ООО Солныш.] -> delete_bymask?filename=*.* (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание – Команда удаляет все подкаталоги из папки Класс01\CLASS\, но файлы не удаляются.

Команда 2

```
Класс01 [Мой компьютер/ООО Солныш.] -> delete_bymask?filename=*.*&onlyfile=true
(124 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание – Команда удаляет все файлы из папки Класс01\CLASS\, но подкаталоги и содержащиеся в них файлы не удаляются.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=delete_bymask?filename=*.*&onlyfile=true
```

5.5 delete_dir

Выражение

```
bool delete_dir(string id)
```

Rest API

```
WebID?method=delete_dir?id=WebID/F:DeletedFolderPath
```

Описание

Удаляет указанный каталог из файловой системы базы данных.

Примечание – Использовать маску вместо имени каталога не допускается, т.е. для удаления группы каталогов команда должна выполняться для каждого каталога отдельно.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена с указанием имени удаляемой папки. Вместе с именем папки допускается указывать относительный путь к папке. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена).

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда 1

```
TestClass [Мой компьютер/ООО Солныш.] ->
delete_dir?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16ED3B01BA25D/F:MyFolder
(46 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание – Команда удаляет подкаталог *MyFolder* в папке *Class*, расположенной в корневой папке класса с идентификатором *C:1D16ED3B01BA25D*.

Команда 2

```
TestClass [Мой компьютер/ООО Солныш.] ->
delete_dir?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16ED3B01BA25D/F:..\MyFol
der (46 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание – Команда удаляет подкаталог *MyFolder* в корневой папке класса с идентификатором *C:1D16ED3B01BA25D*.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=delete_dir?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16ED3B01BA25D/F:
MyFolder
```

5.6 get_dirlist

Выражение

```
XML get_dirlist(string id, string mask)
```

Rest API

```
SourceWebID?method=get_dirlist?mask=FileMask
```

Описание

Возвращает XML-список файлов и каталогов, находящихся в папке *Class*, указанного класса или домена.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, для которого составляется список файлов и каталогов.
mask	строка	Маска имен файлов и каталогов. В маске можно использовать подстановочные символы "?" и "*".

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-список файлов и каталогов.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш./Класс.] -> get_dirlist?mask=* (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<?xml version="1.0"?>
    <FOLDER>
        <DIR Name="backup"/>
        <FILE Size="318" Time="2016-03-25T11:54:29" Name="class.ocl"/>
        <FILE Size="237" Time="2016-02-20T14:43:48" Name="class.oml"/>
        <DIR Name="data"/>
        <DIR Name="index"/>
    </FOLDER>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=delete_dir?mask=*
```

5.7 get_file

Выражение

```
string get_file(string id, string filename, string tc)
```

Rest API

```
SourceWebID?method=get_file?filename=FileName
```

Описание

Команда возвращает полный путь к заданному файлу, расположенному в файловой структуре класса или домена на локальном хосте. Если файл расположен на удаленном хосте, то возвращается полный путь к копии файла в локальном кэше. Если файл в указанном классе (домене) не найден, возвращается сообщение об ошибке, например:

~Error~Файл 'pack.oml' в классе 'TestClass' не найден

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, с указанием имени проверяемого файла. Вместе с именем файла допускается указывать относительный путь к файлу. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена).
filename	строка	Имя проверяемого файла. Вместе с именем файла допускается указывать относительный путь к файлу.
tc	строка	ID транзакции во время получения файла.

Примечание - Имя проверяемого файла можно передать команде двумя способами:

1) через параметр *filename*, при этом имя файла в параметре *id* указывать не надо;

2) указать имя файла в параметре *id*, при этом параметр *filename* использовать не надо.

Дополнительный параметр

```
Не используется.
```

Возвращаемое значение

Тип: строка

Описание: Полный путь к заданному файлу в соответствии с правилами операционной системы.

Пример команды в Windows-клиенте

Команда 1

```
TestClass [Мой компьютер/ООО Солныш./Класс.] -> get_file?filename=data\pack.oml
(124 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
C:\Тестовый хост\d.Work\d.ООО_Солнышко\Класс01\TestClass\CLASS\data\pack.oml
```

Команда 2

```
TestClass [Мой компьютер/ООО Солныш./Класс.] ->
get_file?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3F1E36EDC/F:data\pack.
oml (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
C:\Тестовый хост\d.Work\d.ООО_Солнышко\Класс01\TestClass\CLASS\data\pack.oml
```

Примечание - В первом примере имя проверяемого файла передается в параметре *filename*. Во втором примере имя проверяемого файла передается в параметре *id*.

Команда

```
Здания [oda.support/TEST] -> get_file?filename=data\pack.oml (30 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
C:\ProgramData\ODA\caches\h.000000000000000\d.WORK\d.1CEAD4D04BB28B6\1CD9B0FC2AEC9B4\
data\pack.oml
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_file?filename=data\pack.oml
```

5.8 get_file_path

Выражение

```
string get_file_path(string id, string filename, number base)
```

Rest API

```
SourceWebID?method=get_file_path?filename=FileName[&base=0|1|2|3]
```

Описание

Команда возвращает путь к файлу в соответствии с типом пути, заданном в параметре *base*. Для удаленного хоста эта команда, в отличие от команды *get_file*, возвращает путь к файлу, расположенному непосредственно в файловой структуре сервера, на котором расположен хост, а не к файлу в локальном кэше.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, в котором расположен файл.
filename	строка	Имя файла.
base	число	Тип пути, возвращаемого командой: 0 (по умолчанию) - полный путь, начинающийся с имени диска; 1 - относительный путь от корневого домена; 2 - относительный путь от текущего домена; 3 - относительный путь от класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Путь к файлу в файловой системе сервера с базой данных.

Пример команды в Windows-клиенте

Ниже приведены примеры команды и результаты ее выполнения с разными значениями параметра *base*.

Команда

```
Здания [oda.support/TEST] -> get_file_path?base=0&filename=data\pack.oml (0 мс)
```

Результат выполнения при *base=0*

```
D:\ODABASE\d.Work\d.TEST\Здания\CLASS\data\pack.oml
```

Результат выполнения при *base=1*

```
\d.Work\d.TEST\Здания\CLASS\data\pack.oml
```

Результат выполнения при *base=2*

```
Здания\CLASS\data\pack.oml
```

Результат выполнения при *base=3*

```
data\pack.oml
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:000000000000000/D:WORK/D:1CEAD4D04BB28B6/C:1CD9B0FC2AEC9B4
/?method=get_file_path?base=0&filename=data\pack.oml
```

5.9 load_folder

Выражение

```
string load_folder(string id)
```

Rest API

```
SourceWebID?method=load_folder?id=FullClassID/F:FolderName
```

Описание

Команда загружает из файловой системы класса указанную директорию со всем содержимым в локальный кэш и возвращает на него полный путь.

Необходимые права

RWC

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса с указанием имени копируемого каталога. Если каталог не указан, то копируется каталог *Class* из корневой папки класса. Использовать подстановочные символы "?" и "*" в имени каталога не допускается.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Путь к папке в локальном кэше.

Пример команды в Windows-клиенте

Команда 1

```
Здания [oda.support/TEST] ->
load_folder?id=H:000000000000000/D:WORK/D:1CEAD4D04BB28B6/C:1CD9B0FC2AEC9B4/F:backup
(62 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
C:\ProgramData\ODA\caches\h.000000000000000\d.WORK\d.1CEAD4D04BB28B6\1CD9B0FC2AEC9B4\
backup
```

Команда 2

```
Здания [oda.support/TEST] ->
load_folder?id=H:000000000000000/D:WORK/D:1CEAD4D04BB28B6/C:1CD9B0FC2AEC9B4/F:backup\
2014-02-21\ (31 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
C:\ProgramData\ODA\caches\h.000000000000000\d.WORK\d.1CEAD4D04BB28B6\1CD9B0FC2AEC9B4\
backup\2014-02-21\
```

Примечание – Во втором примере в локальный кэш загружается не весь каталог *backup*, а только содержимое его подкаталога *2014-02-21*. Обратите внимание, что название подкаталога *2014-02-21* обязательно должно заканчиваться символом обратного слеша "\", иначе копирование файлов в кэш не происходит. Если указан только каталог первого уровня как в первом примере (каталог *backup*), то обратный слеш после его имени необязателен.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=load_folder?id=H:000000000000000/D:WORK/D:1CEAD4D04BB28B6/C:1CD9B0FC2AEC9B4/F
:backup
```

5.10 move_file

Выражение

```
bool move_file(string id, string newfilename, string oldfilename, string tc)
```

Rest API

```
SourceWebID?method=move_file?oldfilename=SourceWebID&newfilename=DestinationWebID
```

Описание

Перемещает (переименовывает) указанный файл в файловой структуре базы данных.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, в файловой структуре которого находится перемещаемый файл.

Наименование параметра	Тип	Описание
oldfilename	строка	Имя перемещаемого файла с указанием относительного пути к нему. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена), указанного в параметре *id*.
newfilename	строка	Новое имя перемещаемого файла с указанием нового относительного пути к нему. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена), указанного в параметре *id*.
tc	строка	ID транзакции во время перемещения файла.

Дополнительный параметр

```
Не используется.
```

Возвращаемое значение

Тип: bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш./Класс.] ->
move_file?oldfilename=class.ocl&newfilename=class.new (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
/help.php?method=move_file?oldfilename=http://www.odant.org/api/H:1D03A3F3B5863BB/D:W
ORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E/help.htm&newfilename=http://www.odant.org/api
/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E/index.html
```

5.11 rename_file

Выражение

```
bool rename_file(string id, string filename, string newname)
```

Rest API

```
SourceWebID?method=rename_file?filename=SourceWebID/OldFileName&newname=NewFileName
```

Описание

Команда переименовывает указанный файл в файловой структуре базы данных. Если файл с указанным новым именем уже существует, он будет перезаписан. Если этот файл имеет атрибут *ReadOnly*, то команда выполнена не будет.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена, в файловой структуре которого находится файл.
filename	строка	Относительный путь к файлу, который необходимо переименовать. Относительный путь начинается в папке *Class*, расположенной в корневой папке класса (домена).
newname	строка	Новое имя файла.

Дополнительный параметр

```
Не используется.
```

Возвращаемое значение

Тип: bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш./Класс.] ->
rename_file?filename=class.ocl&newname=class.new (0 мс)
```

Дополнительный параметр

```
Не используется.
```

Результаты выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
/help.php?method=rename_file?filename=http://www.odant.org/api/H:1D03A3F3B5863BB/D:WO
RK/D:1D151E87652DEA0/C:1CCDF1802C35D4E/help.htm&newname=index.html
```

5.12 save_file

Выражение

```
bool save_file(string tc, string id, string filename, number at, number wt, number
```

```
ct)
```

Rest API

```
SourceWebID?method=save_file?id=FullClassID&filename=FileName
```

Описание

Команда копирует указанный файл в файловую структуру базы данных. Если в качестве пункта назначения указан домен, то файл сохраняется в каталоге *domain/class/* этого домена. Если в качестве пункта назначения указан класс, то файл сохраняется в каталоге *class* этого класса. Если в качестве пункта назначения указан конкретный объект класса, то файл сохраняется в каталоге *class/data/files/ИдентификаторОбъекта/* относительно корневой папки класса. Если в папке назначения уже содержится файл с именем копируемого файла, то файл будет перезаписан. Если файл в папке назначения имеет атрибут *ReadOnly*, то он перезаписан не будет, будет создан новый файл с символом тильды «~» в начале имени.

Необходимые права

RWC

Параметры команды

Наименование параметра	Тип	Описание
tc	строка	ID транзакции во время сохранения файла.
id	строка	Полный идентификатор домена, класса или объекта, в который копируются файлы. Примечание - Наличие в классе объекта с указанным идентификатором не проверяется, соответствующая папка создается автоматически и в нее копируется файл.
filename	строка	Имя создаваемого файла.
at	число	Время последнего доступа к файлу.
wt	число	Время изменения файла.
ct	число	Время создания файла.

Примечания

А) Время указывается в количестве наносекунд, прошедших с 1 января 1601г. с учетом поясного времени.

Б) Имя создаваемого файла можно передать команде двумя способами:

- через параметр *filename*, при этом имя файла в параметре *id* указывать не надо;

- указать имя файла в параметре *id*, при этом параметр *filename* использовать не надо.

Дополнительный параметр

Копируемый файл, с указанием пути по правилам операционной системы.

Возвращаемое значение

Тип: bool

Описание:

true – команда выполнена успешно;

false – команда не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда 1

```
TestClass [Мой компьютер/ООО Солныш./Класс.] ->
save_file?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3F1E36EDC/O:1D175463B
ECA104&filename=docum.txt (31 мс)
```

Дополнительный параметр

```
c:\new_docum.txt
```

Результат выполнения

```
true
```

Команда 2

```
TestClass [Мой компьютер/ООО Солныш./Класс.] ->
save_file?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3F1E36EDC/O:1D175463B
ECA104/F:docum.txt (31 мс)
```

Дополнительный параметр

```
c:\new_docum.txt
```

Результат выполнения

```
true
```

Примечание - В первом примере имя проверяемого файла передается в параметре *filename*. Во втором примере имя проверяемого файла передается в параметре *id*.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3F1E36EDC
?method=save_file?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D16BD3F1E36EDC/O:1
D175463BECA104&filename=docum.txt
```

6 Команды работы с пакетами

6.1 get_pack

Выражение

```
XML get_pack(string id)
```

Rest API

```
SourceWebID?method=get_pack?id=ClassID/P:PackName
```

Описание

Команда возвращает XML-описания объектов в указанном пакете (при отсутствии XQUERY-запроса) или результат выполнения XQUERY-запроса над этими объектами. XQUERY-запрос передается в дополнительном параметре.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор пакета.

Дополнительный параметр

XQUERY-запрос для обработки XML-описаний объектов в пакете.

Возвращаемое значение

Тип: XML

Описание: XML-описания объектов (при отсутствии XQUERY-запроса) или результат выполнения XQUERY-запроса над этими объектами.

Пример команды в Windows-клиенте

Команда 1

```
ODANT ->
get_pack?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3/P:2016-03-
21 (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<?xml version="1.0"?>
 <PACK cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" pack="2016-03-21">
 <OBJECT oid="1D1834147E82CE7" Author="1D0EB94C5ACD42E" Owner="1D0EB94C5ACD42E"
cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="2" date="2016-03-21T10:14:17"
cnm="Класс02" size="250" Город="Рязань" Улиц="Крупской" Дом="3" Квар="6"
name="Рязань" update="2016-03-21T10:14:36" user="1D0EB94C5ACD42E"/>
 <OBJECT oid="1D18340EC4F193C" Author="1D0EB94C5ACD42E" Owner="1D0EB94C5ACD42E"
```

```
cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="3" date="2016-03-21T10:11:43"
cnm="Класс02" size="307" Город="Рязань" Улиц="Павлова" Дом="31" name="Рязань"
update="2016-03-21T16:51:11" user="1D0EB94C5ACD42E" Квар="87"/>
 </PACK>
```

Команда 2

```
ODANT ->
get_pack?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3/P:2016-03-
21 (0 мс)
```

Дополнительный параметр

```
document{
    for $a in /PACK/OBJECT group by $a/@oid
        return element Addr {
        $a[1]/(@oid),
        attribute address {$a[1]/(@Город),$a[1]/(@Улиц),$a[1]/(@Дом)}
    }
}
```

Результат выполнения

```
<?xml version="1.0"?>
    <Addr oid="1D1834147E82CE7" address="Рязань Крупской 3"/>
    <Addr oid="1D18340EC4F193C" address="Рязань Павлова 31"/>
```

Примечание - Команда во втором примере отличается от команды в первом примере наличием XQUERY-запроса, переданного в дополнительном параметре. Поэтому результат выполнения команды во втором примере соответствует применению XQUERY-запроса к результатам выполнения команды в первом примере.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_pack?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3/P:20
16-03-21
```

6.2 get_pack_list

Выражение

```
XML get_pack_list(string id)
```

Rest API

```
SourceWebID?method=get_pack_list
```

Описание

Команда возвращает XML-список пакетов (файлов с пакетами), на которые разбит указанный класс. Каждый пакет описывается отдельным тегом <*FILE*>, который содержит три атрибута:

- *Name* – имя файла с пакетом без расширения имени;

- *Ext* – расширение имени файла;

- *Count* – количество объектов в пакете.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-список пакетов (файлов с пакетами).

Пример команды в Windows-клиенте

Команда 1

```
TestClass [Мой компьютер/ООО Солныш./Класс.] -> get_pack_list (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<?xml version="1.0"?>
    <FOLDER>
       <FILE Name="pack" Ext="oml" Count="3"/>
    </FOLDER>
```

Примечание – У класса TestClass режим распределения объектов по пакетам не установлен. Все объекты хранятся в файле *"pack.oml"*, всего в классе 3 объекта.

Команда 2

```
Класс02 [Мой компьютер/ООО Солныш.] -> get_pack_list (15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<?xml version="1.0"?>
    <FOLDER>
        <FILE Name="2016-03-21" Ext="oml" Count="2"/>
        <FILE Name="2015-09-30" Ext="oml" Count="4"/>
        <FILE Name="2015-09-22" Ext="oml" Count="1"/>
    </FOLDER>
```

Примечание – У класса *Класс02* установлен режим распределения объектов по пакетам. Объекты объединяются в пакеты по дате создания. Объекты класса разбиты по трем пакетам.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_pack_list
```

7 Команды работы с хостами

7.1 find_servers

Выражение

```
XML find_servers()
```

Rest API

```
SourceWebID?method=find_servers
```

Описание

Команда осуществляет поиск серверов доступных в данной локальной сети и выдает их список в виде XML-документа. Сервер, выполнивший команду, в этот список не включается.

Необходимые права

Admin

Параметры команды

Отсутствуют

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-список серверов доступных в данной локальной сети.

Пример команды в Windows-клиенте

Команда

```
ODANT -> find_servers (1 с 014 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
<?xml version="1.0"?>
    <SERVERS>
        <SERVER id="1CFD733ACDC38BE" owner="Марочкин М.В." net="N4"
ip="192.168.114.1:211"/>
        <SERVER id="1D0EB94C5ACD42E" owner="Иванов" net="VM1-MAROCHKIN"
ip="192.168.114.139:211"/>
    </SERVERS>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=find_servers
```

7.2 get_config

Выражение

```
XML get_config(string id)
```

Rest API

```
SourceWebID?method=get_config
```

Описание

Команда возвращает XML-описание конфигурации указанного хоста.

Необходимые права

R

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор хоста, XML-описание которого требуется получить.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-описание конфигурации хоста.

Пример команды в Windows-клиенте

Команда

```
ODANT -> get_config?id=H:000000000000000 (140 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
<?xml version="1.0"?>
  <H v="2" i="000000000000000" n="oda.support">
    <D i="ROOT" n="ROOT" s="19" vs="1" l="Корневой домен">
      <C a="2" i="ROOT" n="ROOT" l="Корневой домен" up="2015-10-16T15:16:00"
ab="1">
- // - // - // -
   </H>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_config
```

7.3 get_host_active

Выражение

```
bool get_host_active(string id)
```

Rest API

```
SourceWebID?method=get_host_active
```

Описание

Команда проверяет наличие разрешения на подключение указанному хосту. Разрешить/запретить подключение к хосту можно командой *set_host_active*.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true - соединение с хостом разрешено;

false - соединение с хостом запрещено.

Пример команды в Windows-клиенте

Команда

```
oda.support -> get_host_active (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_host_active
```

7.4 get_host_address

Выражение

```
string get_host_address(string id)
```

Rest API

```
SourceWebID?method=get_host_address?id=H:HostID
```

Описание

Команда выдает сетевой адрес указанного хоста.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор хоста, для которого запрашивается сетевой адрес.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Сетевой адрес хоста.

Пример команды в Windows-клиенте

Команда 1

```
Мой компьютер -> get_host_address (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
127.0.0.1
```

Команда 2

```
ODANT -> get_host_address?id=H:000000000000000 (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
support.oda.su
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_host_address?id=H:000000000000000
```

7.5 get_host_connected

Выражение

```
bool get_host_connected(string id)
```

Rest API

```
SourceWebID?method=get_host_connected?id=H:HostID
```

Описание

Команда проверяет наличие связи с указанным хостом. Связь с хостом может отсутствовать по следующим причинам:

- компьютер пользователя не подключен к сети;
- сервер, на котором установлен хост, выключен или отключен от сети;
- неправильно установлены сетевые параметры хоста;
- на сервере не запущена программа ODANT-сервер.

Примечание – Команда проверяет только наличие физической связи с указанным хостом. Активность соединения проверяется командой *get_host_active*.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор хоста, с которым проверяется связь.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true - связь с хостом установлена;

false - связь с хостом отсутствует.

Пример команды в Windows-клиенте

Команда

```
oda.support -> get_host_connected (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_host_connected?id=H:000000000000000
```

7.6 get_host_name

Выражение

```
string get_host_name(string id)
```

Rest API

```
SourceWebID?method=get_host_name
```

Описание

Команда возвращает имя хоста с указанным идентификатором.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Имя хоста.

Пример команды в Windows-клиенте

Команда

```
ODANT -> get_host_name?id=H:000000000000000 (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
oda.support
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_host_name?id=H:0000000000000
```

7.7 get_is_local

Выражение

```
bool get_is_local(string id)
```

Rest API

```
SourceWebID?method=get_is_local?id=H:HostID
```

Описание

Команда проверяет, что указанный хост является локальным.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true - хост расположен на локальном компьютере;

false - хост расположен на удаленном сервере.

Пример команды в Windows-клиенте

Команда

```
Мой компьютер -> get_is_local (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_is_local?id=H:1D0EB94C5ACD42E
```

7.8 get_publics

Выражение

```
string get_publics(string id)
```

Rest API

```
SourceWebID?method=get_publics?id=H:HostID
```

Описание

Команда выдает список полных идентификаторов классов с публичными методами, расположенных на указанном хосте. Идентификаторы в списке разделены символом ";" (точка с запятой).

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Список полных идентификаторов классов с публичными методами.

Пример команды в Windows-клиенте

Команда

```
oda.support -> get_publics (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
oda://H:000000000000000/D:ROOT/C:000000000000100;
oda://H:000000000000000/D:ROOT/C:000000000000130;
oda://H:000000000000000/D:ROOT/D:1CDF93D135B2826/C:1CDF86EB999F00F;
- // - // - // -
oda://H:000000000000000/D:ROOT/D:1CDA6CD3A64DFDE/C:1CBB7003C4E6126
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_public?id=H:000000000000000
```

7.9 get_remote_hosts_id

Выражение

```
string get_remote_hosts_id(string id)
```

Rest API

```
SourceWebID?method=get_remote_hosts_id?id=H:HostID
```

Описание

Команда возвращает список идентификаторов хостов, подключенных к хосту, указанному в параметре ID. Идентификаторы в списке разделяются символом ";" (точка с запятой).

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Список идентификаторов хостов.

Пример команды в Windows-клиенте

Команда

```
odant.org -> get_remote_hosts_id (0 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
000000000000000;1CC2A7562264BB7;
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_remote_hosts_id
```

7.10 get_starts

Выражение

```
string get_starts(string id)
```

Rest API

```
SourceWebID?method=get_starts?id=H:HostID
```

Описание

Команда выдает список полных идентификаторов классов со стартовыми методами, расположенных на указанном хосте. Идентификаторы в списке разделены символом ";" (точка с запятой).

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Список полных идентификаторов классов со стартовыми методами.

Пример команды в Windows-клиенте

Команда

```
oda.support -> get_starts (15 мс)
```

Дополнительный параметр

Не используется

Результат выполнения

```
oda://H:000000000000000/D:ROOT/D:1CD85D8FBB7A7FB/C:1CBCE795DF3C4D4;
oda://H:000000000000000/D:ROOT/D:1CD85D88BA8F266/C:1CABF8B90BF4132;
- // - // - // -
oda://H:000000000000000/D:ROOT/D:1CDC6FF68FE3424/C:1CD68B37CF6C356;
oda://H:000000000000000/D:ROOT/D:1CDC6FF68FE3424/C:1CD68B37751F24A
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_starts?id=H:000000000000000
```

7.11 get_support_id

Выражение

```
string get_support_id( string id )
```

Rest API

```
SourceWebID?method=get_support_id
```

Описание

Команда возвращает идентификатор хоста, являющегося сервером поддержки для хоста, идентификатор которого указан в параметре *id*.

Необходимые права

R

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор хоста, для которого необходимо узнать идентификатор хоста поддержки.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Полный идентификатор хоста поддержки.

Пример команды в Windows-клиенте

Команда

```
Мой компьютер -> get_support_id (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
/H:000000000000000
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_support_id
```

7.12 get_web_port

Выражение

```
number get_web_port(string id)
```

Rest API

```
SourceWebID?method=get_web_port
```

Описание

Команда возвращает номер порта web-интерфейса указанного хоста.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: число

Описание: Номер порта web-интерфейса.

Пример команды в Windows-клиенте

Команда

```
oda.support -> get_web_port (15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
80
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_web_port
```

7.13 host_restart

Выражение

```
string host_restart( string id, bool force )
```

Rest API

```
SourceWebID?method=host_restart?force=true|false
```

Описание

Команда осуществляет принудительный перезапуск программы ODANT-сервера на указанном хосте. Перезапуск программы может осуществить только пользователь, являющийся администратором соответствующего хоста. Режимом перезапуска ODANT-сервера управляет параметр *force*. Если параметр *force* содержит значение *false*, ODANT-сервер перезапускается только при наличии обновлений. Если параметр *force* содержит значение *true*, ODA-сервер перезапускается независимо от наличия обновлений.

Необходимые права

Admin соответствующего хоста.

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор хоста.
force	bool	Управляет режимом перезапуска ODA-сервера: true – перезапуск осуществляется независимо от наличия обновлений; false (по умолчанию) – перезапуск осуществляется только при наличии обновлений.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Предупреждение об отсутствии прав на выполнение команды.

Пример команды в Windows-клиенте

Команда 1

```
Мой компьютер -> host_restart?force=true (2 с 106 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
no result
```

Команда 2

```
oda.support -> host_restart?force=true (2 с 106 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
~Error~Недостаточно прав у пользователя '1D0EB94C5ACD42E' для перезагрузки сервера.
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=host_restart?force=true
```

7.14 is_host_admin

Выражение

```
bool is_host_admin(string id)
```

Rest API

```
SourceWebID?method=is_host_admin
```

Описание

Команда проверяет, что текущий пользователь является администратором указанного хоста.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор проверяемого хоста.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true – текущий пользователь является администратором хоста;

false – текущий пользователь не является администратором хоста.

Пример команды в Windows-клиенте

Команда

```
Мой компьютер -> is_host_admin (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=is_host_admin
```

7.15 set_host_active

Выражение

```
set_host_active (string id, bool active)
```

Rest API

```
SourceWebID?method=set_host_active?active=true|false
```

Описание

Команда разрешает/запрещает подключение к указанному хосту. Если идентификатор хоста не указан, то команда разрешает/запрещает подключение ко всем доступным хостам включая локальный хост.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор подключаемого/отключаемого хоста.
active	bool	Значение: true – разрешить подключение к хосту; false – запретить подключение к хосту.

Дополнительный параметр

Не используется.

Возвращаемое значение

Отсутствует

Пример команды в Windows-клиенте

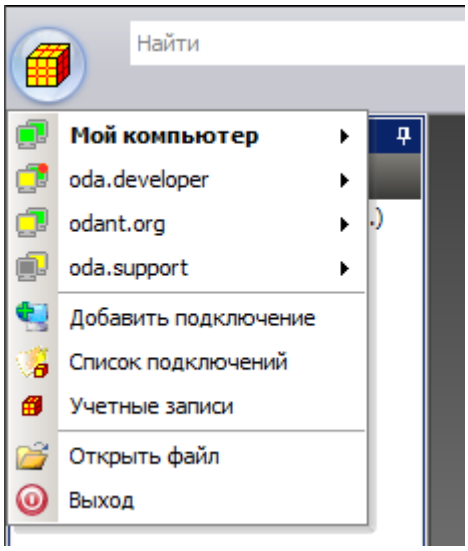
Команда 1

```
oda.support -> set_host_active?active=false (967 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения



На рисунке видно, что хост *oda.support* отключен.

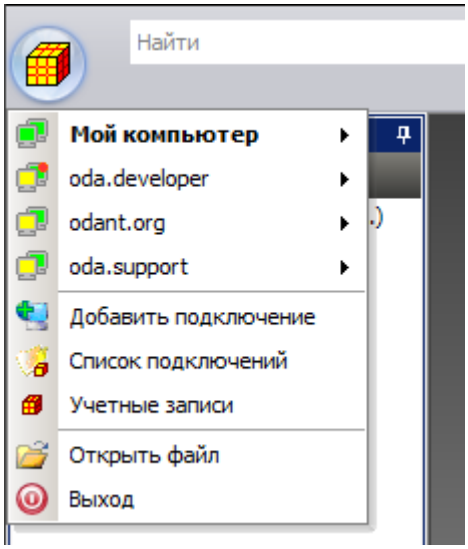
Команда 2

```
oda.support -> set_host_active?active=true (967 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения



На рисунке видно, что хост *oda.support* активен.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=set_host_active?active=true
```

7.16 новая команда xquery

Делает тоже самое, хто и команда xquery_index, но может работать с доменами, классами, объектами, индексами и файлами.

8 Команды работы с пользователями

8.1 get_user_is_online

Выражение

```
bool get_user_is_online(string id)
```

Rest API

```
SourceWebID?method=get_user_is_online?id=H:HostID/U:UserID
```

Описание

Команда возвращает признак, что в текущий момент указанный пользователь находится в сети и подключен к указанному хосту.

Примечание – При входе в систему ODANT пользователь автоматически регистрируется на всех хостах, к которым подключено его рабочее место. Естественно пользователь не может быть зарегистрирован на хостах, связь с которыми отключена в настройках рабочего места или отсутствует по техническим причинам.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Идентификаторы хоста и пользователя в формате: H:HostID/U:UserID

Примечание – При создании хоста ему присваивается идентификатор пользователя-создателя. В текущей версии программы, если в параметре *id* указан идентификатор такого пользователя-создателя и созданный пользователем хост или один из хостов, к которому этот хост подключается, то функция всегда возвращает значение *true*, независимо от реального присутствия пользователя в сети, достаточно, чтобы созданный им хост был активен, а под именем какого пользователя он запущен, не имеет значения. Например, если хост активен и подключен к хосту поддержки с идентификатором H:000000000000000, то команда

```
get_user_is_online?id=H:000000000000000/U:HostId
```

где `HostId` – идентификатор проверяемого хоста,

вернет значение *true*. Аналогично команда

```
get_user_is_online?id=H:HostId/U:HostId
```

возвращает значение *true*.

Исключениями являются хосты oda.support и odant.org. Соответствующие им команды

```
get_user_is_online?id=H:000000000000000/U:000000000000000
get_user_is_online?id=H:1D03A3F3B5863BB/U:1D03A3F3B5863BB
```

всегда возвращают значение *false*.

Дополнительную информацию о хостах смотри в документе «Руководство администратора платформы ODANT» разделы «Подключение хостов» и «Просмотр списка и отключение хостов».

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true - пользователь подключен к хосту;

false - пользователь не подключен к хосту.

Пример команды в Windows-клиенте

Команда

```
ODANT -> get_user_is_online?id=H:000000000000000/U:1CFD733ACDC38BE (78 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Примечание – Пользователь с идентификатором *1CFD733ACDC38BE* в текущий момент находится в сети и подключен к хосту oda.support.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_user_is_online?id=H:000000000000000/U:1CFD733ACDC38BE (78 мс)
```

8.2 get_user_label

Выражение

```
string get_user_label(string id)
```

Rest API

```
SourceWebID?method=get_user_label?id=FullClassID/U:UserID
```

Описание

Команда возвращает имя пользователя с заданным идентификатором. Имя пользователя берется из атрибута *label* тега <*OBJECT*> в XML-описании пользователя. Если атрибут *label* в указанном теге отсутствует, значение берется из атрибута *name*.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Идентификатор пользователя. Примечание – Перед идентификатором пользователя необходимо указать полный идентификатор любого класса или домена из любого подключенного хоста. Поэтому значение, передаваемое через параметр *id*, имеет формат: id=FullClassID/U:UserID

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Имя пользователя.

Пример команды в Windows-клиенте

Команда

```
ODANT -> get_user_label?id=H:000000000000000/D:WORK/U:1D0EB94C5ACD42E (78 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
Иванов
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_user_label?id=H:000000000000000/D:WORK/U:1D0EB94C5ACD42E
```

8.3 get_user_name

Выражение

```
string get_user_name(string id)
```

Rest API

```
SourceWebID?method=get_user_name?id=FullClassID/U:UserID
```

Описание

Команда возвращает имя пользователя с заданным идентификатором. Имя пользователя берется из атрибута *name* тега <*OBJECT*> в XML-описании пользователя.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Идентификатор пользователя. Примечание - Перед идентификатором пользователя необходимо указать полный идентификатор любого класса или домена из любого подключенного хоста. Поэтому значение, передаваемое через параметр *id*, имеет формат: id=FullClassID/U:UserID

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Имя пользователя.

Пример команды в Windows-клиенте

Команда

```
ODANT -> get_user_name?id=H:000000000000000/D:WORK/U:1D0EB94C5ACD42E (78 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
Иванов
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_user_name?id=H:000000000000000/D:WORK/U:1D0EB94C5ACD42E
```

9 Команды работы с объектами

9.1 create_object

Выражение

```
string create_object(number tc, string id)
```

Rest API

```
SourceWebID?method=create_object
```

Описание

Команда создает XML-описание объекта для указанного класса. Созданный объект в базу данных не добавляется. Команда позволяет создать:

- «пустой» объект (шаблон объекта без данных), если не используется дополнительный параметр. Тэг <*OBJECT*> в созданном XML-описании будет содержать только атрибут *oid* с идентификатором объекта и другие стандартные атрибуты, характерные для всех объектов независимо от их класса;

- объект с данными на основании XML-шаблона. Шаблон передается команде через дополнительный параметр. Шаблон должен содержать тэг <*OBJECT*> с атрибутами и дочерними элементами, описывающими индивидуальные свойства объекта. Стандартные атрибуты в тэг <*OBJECT*> можно не добавлять, они автоматически добавляются при создании XML-описания объекта.

Необходимые права

RWC

Параметры команды

Наименование параметра	Тип	Описание
tc	число	ID транзакции во время создания объекта.
id	строка	Полный идентификатор класса, для которого создается XML-описание объекта.

Дополнительный параметр

XML-шаблон создаваемого объекта.

Возвращаемое значение

Тип: строка

Описание: XML-описание объекта

Пример команды в Windows-клиенте

Команда 1

```
Класс02 [Мой компьютер/ООО Солныш.] -> create_object (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<?xml version="1.0"?>
    <OBJECT oid="1D16FCF588B5448" Author="1D0EB94C5ACD42E" Owner="1D0EB94C5ACD42E"
cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="1" date="2016-02-25T16:20:50"
cnm="Класс02" size="192"/>
```

Примечание – Команда создает «пустой» объект класса *Класс02*. Созданному объекту автоматически присвоен идентификатор *1D16FCF588B5448*.

Команда 2

```
Класс02 [Мой компьютер/ООО Солныш.] -> create_object (0 мс)
```

Дополнительный параметр

```
<?xml version="1.0"?>
    <OBJECT Город="Рязань" Улиц="Крупской" Дом="14" Квар="67" name="Рязань"/>
```

Результат выполнения

```
<?xml version="1.0"?>
    <OBJECT Город="Рязань" Улиц="Крупской" Дом="14" Квар="67" name="Рязань"
Author="1D0EB94C5ACD42E" Owner="1D0EB94C5ACD42E" version="1" date="2016-02-
25T16:47:30" bid="1D0F538F5B31E18" cid="1D0F54787BAA8B3" cnm="Класс02" size="192"
oid="1D16FD3120D4ACA"/>
```

Примечание – Команда создает объект класса *Класс02*. В дополнительном параметре в тэге *<OBJECT>* передаются атрибуты, описывающие индивидуальные характеристики объекта.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=create_object
```

9.2 delete_object

Выражение

```
bool delete_object(string id, string tc, bool force)
```

Rest API

```
SourceWebID?method=delete_object[?force=true|false]
```

Описание

Команда удаляет группу объектов из указанного класса. Список идентификаторов удаляемых объектов передается в дополнительном параметре. Идентификаторы в списке разделяются символом пробела. Если указанные объекты доступны только для

чтения, то для их удаления необходимо установить в команде параметр *force* в значение *true*.

Примечание – Объекты, предназначенные только для чтения, содержат в тэге *<OBJECT>* атрибут *ro="True".*

Необходимые права

RWCD

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, из которого удаляются объекты.
tc	число	ID транзакции во время удаления объектов.
force	bool	Значение: true – принудительно удалять объекты с флагом "только чтение"; false (по умолчанию) – не удалять объекты с флагом "только чтение".

Дополнительный параметр

Список идентификаторов удаляемых объектов.

Возвращаемое значение

Тип: bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш.] -> delete_object (15 мс)
```

Дополнительный параметр

```
1D16EFAA97F55B7 1D16EFC8DB7C1F4
```

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=delete_object
```

9.3 delete_object_by

Выражение

```
bool delete_object_by(string id, string tc, bool force)
```

Rest API

```
SourceWebID?method=delete_object_by[?force=true|false]
```

Описание

Команда выбирает и удаляет из указанного класса объекты, удовлетворяющие заданному условию. Условие передается в дополнительном параметре. Если класс содержит объекты доступные только для чтения, то для их удаления необходимо установить в команде параметр *force* в значение *true*.

Примечание – Объекты, предназначенные только для чтения, содержат в тэге *<OBJECT>* атрибут *ro="True".*

Необходимые права

RWCD

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, из которого удаляются объекты.
tc	строка	ID транзакции, во время удаления объектов.
force	bool	Значение: true – принудительно удалять объекты с флагом "только чтение"; false (по умолчанию) – не удалять объекты с флагом "только чтение".

Дополнительный параметр

Условие, которому должны соответствовать удаляемые объекты. Условие соответствует предикату языка XPath. При записи условия квадратные скобки, в которые заключается предикат в языке XPath, использовать не надо.

Возвращаемое значение

Тип: bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш.] -> delete_object_by (15 мс)
```

Дополнительный параметр

```
@Year=1941
```

Результат выполнения

```
True
```

Примечание – Команда удаляет все объекты, у которых поле *Year* содержит значение *1941*.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=delete_object_by
```

9.4 find_objects_id

Выражение

```
string find_objects_id( string id )
```

Rest API

```
SourceWebID?method=find_objects_id
```

Описание

Команда выдает список идентификаторов объектов указанного класса, удовлетворяющий условию, переданному в дополнительном параметре. Идентификаторы объектов в списке разделяются пробелами.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, в котором происходит поиск объектов.

Дополнительный параметр

Условие, которому должны соответствовать искомые объекты. Условие является предикатом языка XPath. При записи условия квадратные скобки, в которые заключается предикат в языке XPath, использовать не надо.

Возвращаемое значение

Тип: строка

Описание: Список идентификаторов найденных объектов.

Пример команды в Windows-клиенте

Команда

```
Класс02 [Мой компьютер/ООО Солныш.] -> find_objects_id (15 мс)
```

Дополнительный параметр

```
@Город="Рязань"
```

Результат выполнения

```
1D0FB7AC89E38E5 1D1834147E82CE7 1D0834047D82124
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=find_objects_id
```

9.5 generate_id

Выражение

```
string generate_id(string id, number count)
```

Rest API

```
SourceWebID?method=generate_id?count=NumberOfIdentifiers
```

Описание

Команда генерирует и выдает заданное количество уникальных идентификаторов. Идентификаторы в списке разделяются символом пробела.

Необходимые права

RWC

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор элемента структуры базы данных.

Наименование параметра	Тип	Описание
count	число	Количество уникальных идентификаторов, которое необходимо сгенерировать. По умолчанию генерируется один идентификатор.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Список уникальных идентификаторов.

Пример команды в Windows-клиенте

Команда

```
Класс01 [Мой компьютер/ООО Солныш.] -> generate_id?count=4 (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
1D1706573996F79 1D1706573996F7A 1D1706573996F7B 1D1706573996F7C
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=generate_id?count=4
```

9.6 get_childs_id

Выражение

```
string get_childs_id(string id)
```

Rest API

```
SourceWebID?method=get_childs_id
```

Описание

Команда выдает список идентификаторов дочерних классов. Идентификаторы в списке разделены символом ";" (точка с запятой).

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор родительского класса.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: строка

Описание: Список идентификаторов дочерних классов.

Пример команды в Windows-клиенте

Команда

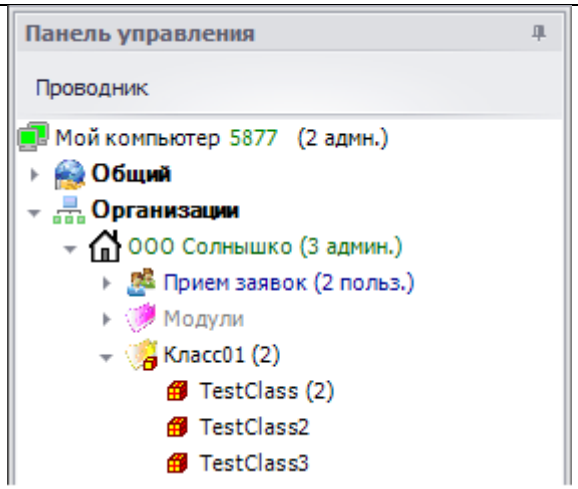
```
Класс01 [Мой компьютер/ООО Солныш.] -> get_childs_id (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
1D16BD3F1E36EDC;1D186894BDDF2FB;1D18689F73EA555
```



Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_childs_id
```

9.7 get_object

Выражение

```
string get_object(string id, string tc)
```

Rest API

```
SourceWebID?method=get_object?id=WebID/O:ObjectID
```

Описание

Команда выдает XML-описание указанного объекта (при отсутствии XQUERY-запроса) или результат выполнения XQUERY-запроса над этим XML-описанием. XQUERY-запрос передается в дополнительном параметре.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор объекта.
tc	строка	ID транзакции во время обработки объекта.

Дополнительный параметр

XQUERY-запрос для обработки XML-описания объекта.

Возвращаемое значение

Тип: Строка

Описание: XML-описание объекта (XQUERY-запрос не задан) или результат выполнения XQUERY-запроса к XML-описанию объекта.

Пример команды в Windows-клиенте

Команда 1

```
ODANT ->
get_object?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3/O:1D18053D
FFD9A0B (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<?xml version="1.0"?>
 <OBJECT oid="1D18053DFFD9A0B" Author="1D0EB94C5ACD42E" Owner="1D0EB94C5ACD42E"
cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="4" date="2016-03-17T16:49:49"
cnm="Класс02" size="303" Город="Рязань" Улиц="Ленина" Дом="22" Квар="45"
name="Рязань" update="2016-03-28T12:58:53" user="1D0EB94C5ACD42E"/>
```

Команда 2

```
ODANT ->
get_object?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3/O:1D18053D
FFD9A0B (0 мс)
```

Дополнительный параметр

```
document{
    element Accommodation {
        OBJECT/(@Город),OBJECT/(@Улиц),OBJECT/(@Дом)
    }
}
```

Результат выполнения

```
<?xml version="1.0"?>
    <Accommodation Город="Рязань" Улиц="Ленина" Дом="22"/>
```

Примечание - Команда во втором примере отличается от команды в первом примере наличием XQUERY-запроса, переданного в дополнительном параметре. Поэтому результат выполнения команды во втором примере соответствует выполнению XQUERY-запроса над результатами выполнения команды в первом примере.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_object?id=H:1D0EB94C5ACD42E/D:WORK/D:1D0F538F5B31E18/C:1D0F54787BAA8B3/O:
1D18053DFFD9A0B (0 мс)
```

9.8 get_object_class

Выражение

```
GID get_object_class(string id)
```

Rest API

```
SourceWebID?method=get_object_class?id=WebID/O:ObjectID
```

Описание

Команда возвращает идентификатор класса, которому принадлежит указанный объект.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор объекта.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: GID

Описание: Идентификатор класса, которому принадлежит указанный объект.

Пример команды в Windows-клиенте

Команда

```
ODANT ->
get_object_class?id=H:000000000000000/D:WORK/D:1CEAD4D04BB28B6/C:1CD9B0FC2AEC9B4/O:1CEC64710CB2BF2 (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
1CD9B0FC2AEC9B4
```

Примечание – В качестве примера взят объект класса *Здания* в домене-организации *TEST* хоста *oda.support*.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:000000000000000/D:WORK/D:1CEAD4D04BB28B6/C:1CD9B0FC2AEC9B4?method=get_object_class?id=H:000000000000000/D:WORK/D:1CEAD4D04BB28B6/C:1CD9B0FC2AEC9B4/O:1CEC64710CB2BF2 (0 мс)
```

9.9 get_objects_list

Выражение

```
XML get_objects_list(string id)
```

Rest API

```
SourceWebID?method=get_objects_list
```

Описание

Команда выдает XML-описания объектов указанного класса по заданному списку идентификаторов объектов. Список идентификаторов передается в дополнительном параметре. Идентификаторы в списке должны быть разделены пробелом.

Необходимые права

R

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, из которого извлекаются объекты.

Дополнительный параметр

Список идентификаторов объектов.

Возвращаемое значение

Тип: XML

Описание: XML-список объектов.

Пример команды в Windows-клиенте

Команда

```
Класс02 [Мой компьютер/ООО Солныш.] -> get_objects_list (15 мс)
```

Дополнительный параметр

```
1D0FB7AC89E38E5 1D18053DFFD9A0B
```

Результат выполнения

```
<?xml version="1.0"?>
    <OBJECTS>
        <OBJECT oid="1D18053DFFD9A0B" Author="1D0EB94C5ACD42E"
Owner="1D0EB94C5ACD42E" cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="2"
date="2016-03-17T16:49:49" cnm="Класс02" size="250" Город="Рязань" Улиц="Ленина"
Дом="22" Квар="45" name="Рязань" update="2016-03-17T16:57:56"
user="1D0EB94C5ACD42E"/>
        <OBJECT oid="1D0FB7AC89E38E5" Author="1D0EC800AFBA05B"
Owner="1D0EC800AFBA05B" cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="4"
date="2015-09-30T15:23:16" cnm="Класс02" size="305" Город="Рязань" Улиц="Крупской"
Дом="14" Квар="104" name="Рязань" update="2015-10-08T15:34:24"
user="1D0EC800AFBA05B"/>
    </OBJECTS>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=get_objects_list
```

9.10 is_has_childs

Выражение

```
bool is_has_childs(string id)
```

Rest API

```
SourceWebID?method=is_has_childs
```

Описание

Команда проверяет наличие дочерних элементов у класса или домена.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true – имеет дочерние элементы;

false – не имеет дочерних элементов.

Пример команды в Windows-клиенте

Команда

```
TEST [oda.support] -> is_has_childs (15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:000000000000000/D:WORK/D:1CEAD4D04BB28B6?method=is_has_childs
```

9.11 run_method

Выражение

```
bool run_method(string id, string method)
```

Rest API

```
SourceWebID?method=run_method?method=MethodName
```

Описание

Команда запускает на выполнение метод с заданным именем на стороне сервера и возвращает признак успешности запуска.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Идентификатор класса, которому принадлежит метод.
method	строка	Имя метода.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: bool

Описание:

true – запуск метода выполнен успешно;

false – запуск метода не выполнен, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
Печатные шаблоны [Мой компьютер/ООО Солньш./Печат.бланк.] ->
run_method?method=Print (0 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
true
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=run_method?method=Print
```

9.12 save_object

Выражение

```
GID save_object(string id, string tc)
```

Rest API

```
SourceWebID?method=save_object
```

Описание

Команда сохраняет в указанном классе объект, преданный через дополнительный параметр в виде XML-описания. При сохранении проверяется идентификатор объекта.

Если объект с указанным идентификатором уже существует в классе, то он будет обновлен. Если объект с указанным идентификатором отсутствует в классе, то будет создан новый объект. Если в XML-описании отсутствует идентификатор объекта, то создается новый объект с автоматически-присвоенным уникальным идентификатором. Если XML-описание объекта отсутствует в дополнительном параметре, команда создает объект, в котором все поля пустые, в том числе и те, которым значение должно присваиваться по умолчанию. Если в XML-описании объекта присутствует несколько тэгов <*OBJECT*>, то объект создается (обновляется) только из первого тэга, остальные теги игнорируются.

Необходимые права

RWC

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, которому принадлежит объект.
tc	строка	ID транзакции во время сохранения объекта.

Дополнительный параметр

XML-описание сохраняемого объекта.

Возвращаемое значение

Тип: GID

Описание: Идентификатор сохраненного объекта.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солньш./Класс.] -> save_object (15 мс)
```

Дополнительный параметр

```
<?xml version="1.0"?>
    <OBJECT cid="1D16BD3F1E36EDC" bid="1D0F538F5B31E18" cnm="TestClass"
Фамил="Иванов" Имя="Иван" Отчес="Иванович"/>
```

Результат выполнения

```
1D17480C433F078
```

Примечание – Команда автоматически присваивает объекту уникальный идентификатор и создает новый объект в классе TestClass.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=save_object
```

9.13 save_objects

Выражение

```
bool save_objects(string tc, string id)
```

Rest API

```
SourceWebID?method=save_objects
```

Описание

Команда сохраняет в указанном классе объекты, преданные через дополнительный параметр в виде XML-документа. При сохранении проверяются идентификаторы объектов. Если объект с указанным идентификатором уже существует в классе, то он будет обновлен. Если объект с указанным идентификатором отсутствует в классе, то будет создан новый объект. Если в XML-описании отсутствует идентификатор объекта, то создается новый объект с автоматически-присвоенным уникальным идентификатором. Если XML-описание объектов отсутствует в дополнительном параметре, команда выдает сообщение об ошибке:

~Error~Ошибка в загрузке xml-списка объектов.

Необходимые права

RWC

Параметры команды

Наименование команды	Тип	Описание
tc	строка	ID транзакции во время сохранения объектов.
id	строка	Полный идентификатор класса, которому принадлежат объекты.

Дополнительный параметр

XML-документ со списком объектов.

Возвращаемое значение

Тип: bool

Описание:

true – операция выполнена успешно;

false – операция не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш./Класс.] -> save_objects (15 мс)
```

Дополнительный параметр

```
<?xml version="1.0"?>
    <PACK>
        <OBJECT cid="1D16BD3F1E36EDC" bid="1D0F538F5B31E18" cnm="TestClass"
Фамил="Иванов" Имя="Иван" Отчес="Иванович"/>
        <OBJECT cid="1D16BD3F1E36EDC" bid="1D0F538F5B31E18" cnm="TestClass"
Фамил="Петров" Имя="Петр" Отчес="Петрович"/>
    </PACK>
```

Результат выполнения

```
true
```

Примечание – Команда автоматически присваивает обоим объектам уникальные идентификаторы и создает два новых объекта в классе TestClass.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=save_objects
```

9.14 save_objects_files

Выражение

```
string save_objects_files(string tc, string id)
```

Rest API

```
SourceWebID?method=save_objects_files
```

Описание

Команда извлекает из указанного архива все файлы и сохраняет их в файловой структуре указанного класса. При извлечении и копировании файлов структура папок, сохраненная в архиве, не изменяется. Архив должен иметь формат ZIP. Файлы сохраняются в каталоге *class/data/files/* относительно корневой папки класса. Имена файлов в папке назначения могут совпадать с именами файлов в архиве, в этом случае они будут перезаписаны файлами из архива. Если файлы в папке назначения имеют атрибут *ReadOnly*, то они перезаписаны не будут.

Необходимые права

RWC

Параметры команды

Наименование параметра	Тип	Описание
tc	строка	ID транзакции во время сохранения файлов.

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, в который копируются файлы.

Дополнительный параметр

Путь к файлу архива в формате ZIP, из которого копируются файлы. Путь указывается по правилам файловой системы операционной системы.

Возвращаемое значение

Тип: bool

Описание:

true – команда выполнена успешно;

false – команда не выполнена, при этом дополнительно возвращаются код ошибки и ее описание.

Примечание – В текущей версии платформы при успешном выполнении команда возвращает значение «no result».

Пример команды в Windows-клиенте

Команда

```
TestClass [Мой компьютер/ООО Солныш.] -> save_objects_files (78 мс)
```

Дополнительный параметр

```
c:\archive.zip
```

Результат выполнения

```
no result
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=save_objects_files
```

9.15 search_oids

Выражение

```
string search_oids(string id, string mask, bool allwords)
```

Rest API

```
SourceWebID?method=search_oids?mask=PacketMask
```

Описание

Команда выдает список идентификаторов объектов, содержащих подстроку поиска, переданную в дополнительном параметре. Идентификаторы в списке разделены

символами пробела. Поиск подстроки осуществляет в значениях атрибутов объектов. Поиск осуществляется в XML-описании объектов среди значений всех атрибутов, в том числе и среди служебных атрибутов таких, как *oid*, *user*, *date*, *update* и т.д. Если объекты класса разделены на пакеты, то поиск осуществляется только в пакетах, соответствующих маске, указанной в параметре *mask*.

Дополнительные сведения:

1) Поиск подстроки осуществляется без учета регистра символов.

2) Если искомая подстрока содержит знаки пробелов, то ее необходимо заключать в двойные или одинарные кавычки.

3) Дополнительный параметр может содержать несколько подстрок, разделенных пробелами. В этом случае правила отбора объектов определяется значением параметра *allwords*.

4) Для составления сложных условий отбора, в дополнительном параметре можно использовать знаки логических операций, которые ставятся между искомыми подстроками. Искомые подстроки должны отделяться пробелами от знаков логических операций. Операторы **или**, **or**, **|** (вертикальная черта) соответствуют логической операции «ИЛИ». Операторы **и**, **and**, **&** соответствуют логической операции «И». Если искомые подстроки разделены только пробелами, то логические операции между ними определяются параметром *allwords*. По правилам логики операция «И» имеет более высокий приоритет (выполняется раньше), чем операция «ИЛИ», чтобы изменить порядок выполнения логических операций необходимо использовать круглые скобки.

5) Если объектов удовлетворяющих условию поиска не найдено, то команда выдает сообщение:

no result

Необходимые права

R

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, в котором происходит поиск объектов.
mask	строка	Маска имен файлов с пакетами объектов, в которых осуществляется поиск. Этот параметр используется только для классов, у которых установлен режим распределения объектов по пакетам. В этом режиме для хранения объектов класса используются несколько файлов, каждому пакету соответствует отдельный файл. В маске допускается использовать стандартные подстановочные символы для имен файлов: "*" и "?". По умолчанию в качестве маски используется подстановочный символ "*", что соответствует поиску во всех файлах (во всех пакетах объектов).

Наименование параметра	Тип	Описание
		Пример: Пусть объекты распределены по пакетам по дате создания, тогда команда с параметром *loadmask=2015** будет работать только с объектами, созданными в 2015 году.
allwords	bool	Используется, если между соседними подстроками отсутствует логический оператор. В этом случае параметр указывает, какой логической операции соответствует пробел, связывающий подстроки. Значение: *true* – пробел соответствует логической операции "И"; *false* (по умолчанию) – пробел соответствует логической операции "ИЛИ".

Дополнительный параметр

Искомые подстроки, соединенные знаками логических операций или пробелами.

Возвращаемое значение

Тип: строка

Описание: Список идентификаторов объектов, соответствующих условию поиска.

Пример команды в Windows-клиенте

Команда

```
Класс02 [Мой компьютер/ООО Солныш.] -> search_oids?mask=2015* (0 мс)
```

Дополнительный параметр

```
-09
```

Результат выполнения

```
1D0F5480DF0FEE6 1D0FB7AC89E38E5
```

Примечание – Объекты класса разбиты по пакетам. Объекты объединены в пакеты по дате создания. Параметр *mask=2015** указывает, что поиск осуществляется среди объектов, созданных в 2015 году.

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1CCDF1802C35D4E
?method=search_oids?mask=PacketMask
```

9.16 reculc

10 Команды работы с архивом

10.1 get_backup_class

Выражение

```
XML get_backup_class(string d, number i, string id)
```

Rest API

```
SourceWebID?method=get_backup_class?d=Date&i=VersionNumber
```

Описание

Команда возвращает сохраненную в архиве версию XML-описания класса или домена. В параметре **d** указывается дата, когда описание класса (домена) было помещено в архив. В параметре **i** указывается порядковый номер описания класса (домена) в архиве на дату **d**. Список доступных значений параметров **d** и **i** возвращается командой *get_backups_info*.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса или домена.
d	строка	Дата записи описания класса (домена) в архив. Дата имеет формат YYYY-MM-DD, где YYYY - год, MM - номер месяца, DD - день.
i	число	Порядковый номер описания класса (домена) в архиве на дату d.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-описание класса или домена, считанное из архива.

Пример команды в Windows-клиенте

Команда

```
Класс02 [Мой компьютер/ООО Солныш.] -> get_backup_class?i=2&d=2016-03-21 (15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<CLASS Author="1D0EC800AFBA05B" ClassId="1D0F54787BAA8B3" Date="2015-09-
22T18:01:16" Name="Класс02" Label="Класс02" Parent="1D0F538F5B31E18" Update="2015-
10-06T16:56:05" Backup="True">
<SF/>
    <METADATA>
    <ATTR Name="Город" Label="Город" List="False"/>
    <ATTR Name="Улиц" Label="Улица" List="False"/>
    <ATTR Name="Дом" Label="Дом" List="False"/>
    <ATTR Name="Квар" Label="Квартира" List="False"/>
    </METADATA>
</CLASS>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1D0F54787BAA8B3
?method=get_backup_class?i=2&d=2016-03-21
```

10.2 get_backups_info

Выражение

```
XML get_backups_info(string id, string mask, number types)
```

Rest API

```
SourceWebID?method=get_backups_info?mask=DateMask&types=TypesMask
```

Описание

Команда выдает XML-список изменений в указанном классе или домене. В параметре *types* передается битовая маска запрашиваемых типов изменений. В параметре *mask* передается маска дат, на которые ищутся изменения. Параметр *mask* всегда необходимо указывать явно, без него команда не работает. По умолчанию параметр *types* имеет значение 0, при этом в XML-списке указываются только даты произведенных изменений, вся остальная информация отсутствует.

Примечание - Информация обо всех изменениях в классе и домене хранится в соответствующих подкаталогах папки *CLASS/backup*, расположенной в корневой папке класса.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор домена или класса.
mask	строка	Маска даты изменения домена, класса или объекта. В маске можно использовать подстановочные символы "*" и "?". Дата имеет формат YYYY-MM-DD, где YYYY – год, MM - номер месяца, DD - день.
types	число	Битовая маска типов запрашиваемых изменений (1 – изменение объектов, 2 – удаление объектов, 4 – изменение параметров класса, 8 – изменение/удаление файлов).

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-список изменений в указанном классе или домене.

Пример команды в Windows-клиенте

Команда

```
Класс02 [Мой компьютер/ООО Солныш.] -> get_backups_info?types=15&mask=2016* (109
мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<DATASET>
    <METADATA>
        <ATTR Name='d' Label='Дата' Type='ДатаВремя' Format='ShortDate'/>
        <ATTR Name='k' Label='Тип' Type='Строка' ValueList='"1=изменение
объекта","2=удаление объекта","4=изменение класса","8=изменение/удаление файла"'/>
        <ATTR Name='u' Label='Пользователь' Type='User'/>
        <ATTR Name='ip' Label='IP'/>
        <ATTR Name='t' Label='Время'/>
        <ATTR Name='o' Label='OID'/>
        <ATTR Name='n' Label='Имя'/>
    </METADATA>
    <DATA>
        <R d='2016-03-17' k='1' i='1' o='1D18053DFFD9A0B' n='Рязань'
u='1D0EB94C5ACD42E' t='16:57:56'/>
        <R d='2016-03-21' k='4' i='1' u='1D0EB94C5ACD42E' ip='127.0.0.1'
t='09:47:37'/>
        <R d='2016-03-21' k='4' i='2' u='1D0EB94C5ACD42E' ip='127.0.0.1'
t='09:57:51'/>
```

```
        <R d='2016-03-23' k='8'
n='data\files\1D0F5480DF0FEE6\230303500828561.sys'/>
        <R d='2016-03-23' clr='red' k='2' i='1' o='1D0F5480DF0FEE6' n='Рязань'
u='1D0EB94C5ACD42E' t='16:32:32'/>
    </DATA>
</DATASET>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1D0F54787BAA8B3
?method=get_backups_info?types=15&mask=2016*
```

10.3 get_backup_deleted_object

Выражение

```
XML get_backup_deleted_object(string id, string d, string i)
```

Rest API

```
SourceWebID?method=get_backup_deleted_object?d=Date&i=Version
```

Описание

Команда возвращает XML-описание объекта, который был удален. В параметре **d** указывается дата, когда описание объекта было помещено в архив. В параметре **i** указывается порядковый номер описания объекта в архиве на дату **d**. Список доступных значений параметров **d** и **i** возвращается командой *get_backups_info*.

Примечание - При записи объектов в архив на определенную дату используется сквозная нумерация сохраненных объектов независимо от их идентификаторов, и команда извлечения объектов из архива не учитывает идентификаторы объектов. Поэтому для извлечения конкретного объекта с определенным идентификатором необходимо предварительно узнать его порядковый номер в архиве по данным, возвращаемым командой *get_backups_info*.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, которому принадлежит объект.
d	строка	Дата записи объекта в архив. Дата имеет формат YYYY-MM-DD, где YYYY – год, MM – номер месяца, DD – день.
i	строка	Порядковый номер объекта в архиве на дату **d**.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-описание объекта, удаленного датой **d**, хранящегося в архиве под номером **i**.

Пример команды в Windows-клиенте

Команда

```
Класс02 [Мой компьютер/ООО Солныш.] -> get_backup_deleted_object?i=2&d=2016-03-25
(31 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<OBJECT oid="1D18053DFFD9A0B" Author="1D0EB94C5ACD42E" Owner="1D0EB94C5ACD42E"
cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="5" date="2016-03-11T16:49:49"
cnm="Класс02" size="250" Город="Рязань" Улиц="Ленина" Дом="22" Квар="45"
name="Рязань" update="2016-03-24T16:57:56" user="1D0EB94C5ACD42E"/>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1D0F54787BAA8B3
?method=get_deleted_object?i=2&d=2016-03-21
```

10.4 get_backup_update_object

Выражение

```
XML get_backup_update_object(string id, string d, string i)
```

Rest API

```
SourceWebID?method=get_backup_update_object?d=Date&i=Version
```

Описание

Команда возвращает сохраненную в архиве версию XML-описания объекта. В параметре **d** указывается дата, когда описание объекта было помещено в архив. В параметре **i** указывается порядковый номер описания объекта в архиве на дату **d**. Список доступных значений параметров **d** и **i** возвращается командой *get_backups_info*.

Примечание – При записи объектов в архив на определенную дату используется сквозная нумерация сохраненных объектов независимо от их идентификаторов, и команда извлечения объектов из архива не учитывает идентификаторы объектов. Поэтому для извлечения конкретной версии объекта с определенным идентификатором

необходимо предварительно узнать ее порядковый номер в архиве по данным, возвращаемым командой *get_backups_info*.

Необходимые права

Admin

Параметры команды

Наименование параметра	Тип	Описание
id	строка	Полный идентификатор класса, которому принадлежит объект.
d	строка	Дата записи объекта в архив. Дата имеет формат YYYY-MM-DD, где YYYY - год, MM - номер месяца, DD - день.
i	строка	Порядковый номер объекта в архиве на дату **d**.

Дополнительный параметр

Не используется.

Возвращаемое значение

Тип: XML

Описание: XML-описание предыдущей версии объекта, хранящейся в архиве под номером **i** на дату **d**.

Пример команды в Windows-клиенте

Команда

```
Класс02 [Мой компьютер/ООО Солныш.] -> get_backup_object?i=2&d=2016-03-21 (15 мс)
```

Дополнительный параметр

```
Не используется
```

Результат выполнения

```
<OBJECT oid="1D18053DFFD9A0B" Author="1D0EB94C5ACD42E" Owner="1D0EB94C5ACD42E"
cid="1D0F54787BAA8B3" bid="1D0F538F5B31E18" version="4" date="2016-03-11T16:49:49"
cnm="Класс02" size="250" Город="Рязань" Улиц="Ленина" Дом="22" Квар="45"
name="Рязань" update="2016-03-17T16:57:56" user="1D0EB94C5ACD42E"/>
```

Пример команды в Web-браузере

```
http://www.odant.org/api/H:1D03A3F3B5863BB/D:WORK/D:1D151E87652DEA0/C:1D0F54787BAA8B3
?method=get_backup_object?i=2&d=2016-03-21
```