

S-100 – Часть 13

Скриптинг

От автора перевода: В данной части применяются термины, которые нашли широкое применение, но не имеют адекватных терминов в русском языке. Значение или определение таких терминов дается по тексту в данной части, либо в глоссарии терминов в заключительной части стандарта S-100. Например, скриптинг, парсинг, хостинг и т.п.

Содержание

13-1 Цель	3
13-2 Соответствие	3
13-3 Нормативные ссылки	3
13-4 Назначение	3
13-5 Каталог сценариев	3
13-5.1 Дистрибуция	5
13-5.2 Специфические функции каталога скриптинга домена	5
13-6 Обмен данными	5
13-6.1 Схема DEF	5
13-6.1.1 Специальные символы	6
13-6.1.2 Кодирование строк	6
13-6.1.3 Парсинг	7
13-6.2 Путь к атрибуту	8
13-7 Требования к хостингу	8
13-7.1 Версия Lua	9
13-7.2 Кодирование символов	9
13-7.3 Обработка ошибок	9
13-7.4 Параметры массивов	9
13-7.5 Функции хостов	9
13-7.5.1 Совместимость	10
13-8 Функции стандартных скриптов	10
13-8.1 Стандартные функции каталога	12
13-8.1.1 Функции создания объектов	12
13-8.1.2 Функции создания типа информации	22
13-8.1.3 Другие функции	32
13-8.2 Стандартные функции хоста	33
13-8.2.1 Функции доступа к данным	34
13-8.2.2 Функции доступа к типу информации	41
13-8.2.3 Функции пространственных операций	46
13-8.2.4 Функции поддержки отладки	47

13-1 Цель

Эта часть определяет стандартный механизм для включения поддержки сценариев (скриптинга) в продуктах на основе S-100. Скриптинг обеспечивает обработку наборов данных, созданных на основе S-100, с помощью файлов сценариев, написанных на языке программирования Lua.

13-2 Соответствие

Скрипты, соответствующие этой части, реализуются с использованием версии 5.1 языка программирования Lua.

13-3 Нормативные ссылки

Для применения этого документа требуется знание следующих нормативных документов. Применительно к датированным ссылкам требуется знать только упомянутое издание. Для недатированных ссылок применяется последнее издание нормативного документа (включая поправки).

Lua 5.1 Reference Manual, <https://www.lua.org/manual/5.1/>

ISO 19125-1:2004, *Geographic information -- Simple feature access -- Part 1: Common architecture*

13-4 Назначение

Эта часть предназначена для нормативного выражения и обработки правил для продуктов на базе S-100. К возможным примерам использования относятся: правила изображения, правила эксплуатационной совместимости продуктов, правила определения опасностей для судоходства, правила проверки данных и т.д.

Использование скриптов устраняет двусмысленность выражения правила, обеспечивает согласованность между приложениями и позволяет изменять правила или расширять их с помощью обновлений каталога.

13-5 Каталог сценариев

Каталог сценариев (рисунок 1) - это коллекция файлов сценариев, написанных для использования в домене сценариев.

Например, изображение - это домен сценариев. Файлы правил, содержащиеся в каталоге изображений Lua, включают каталог сценариев.

Все каталоги сценариев гарантированно содержат стандартные функции каталога, определенные в пункте 13-8.1. Каталоги сценариев могут дополнительно содержать функции каталогов конкретного домена. Стандартные функции каталога упрощают создание, интеграцию и тестирование скриптов в домене скриптов.



Рисунок 13-1 – Состав каталога сценариев

Для применения правил в домене сценариев каталоги сценариев взаимодействуют с функциями хоста. Связь между каталогом сценариев и функциями хоста показана на рисунке 13-2 ниже. Функции хоста позволяют отделить каталог сценариев от реализации хостом концепций и функций S-100.



Рисунок 13-2 – Взаимодействие Каталога сценариев и Хоста внутри домена сценариев

13-5.1 Дистрибуция

Механизм дистрибуции каталога скриптинга определяется в домене сценариев. Например, S-100 Часть 9А включает в себя каталог скриптинга в рамках каталога изображений; распространение каталога скриптинга осуществляется через распространение каталога изображений.

Каждая реализация каталога скриптинга должна включать все стандартные функции каталога.

13-5.2 Специфические функции каталога скриптинга домена

Стандартные скриптовые функции всегда доступны в каталоге скриптинга. Части S-100, использующие скриптинг, могут предоставлять дополнительные скриптовые функции, необходимые для поддержки специфической для домена функциональности. В этом случае дополнительные функции называются "специфическими функциями домена".

Специфические функции домена, предназначенные для взаимодействия с каталогом хоста/ сценариев (рисунок 2), должны быть указаны в соответствующей части S-100. В S-100 нет необходимости указывать специфические функции домена, используемые внутри каталога скриптинга.

Например, предположим, что S-100 Часть N использует скрипты и требует добавления функций скриптов X, Y и Z. Если функции X и Y вызываются из хоста, но функция Z вызывается только из функций X и Y, S-100 Часть N должна указать требуемые функции X и Y и предоставить документацию для каждой функции. Поскольку функция Z используется только внутри каталога скриптинга, ее не нужно документировать.

Специфические функции домена, используемые для взаимодействия между хостом и каталогом скриптинга, называются "специфическими функциями хоста домена" или "специфическими функциями каталога скриптинга домена" в зависимости от того, реализуются ли они хостом или внутри каталога скриптинга.

13-6 Обмен данными

Данные, передаваемые хосту из каталога скриптинга, могут быть получены с помощью функций Lua C API, соответствующих типу данных. Для простых типов данных, таких как nil, boolean, string и number, поиск данных тривиален. Для более сложных типов данных каталог скриптинга кодирует данные с использованием формата обмена данными (DEF), описанного в этом разделе.

13-6.1 Схема DEF

Формат обмена данными (DEF) – это строка, отформатированная как описано ниже. Хост-парсинг DEF прост в реализации с использованием возможностей парсинга, встроенных во все популярные языки программирования. Хост-парсинг DEF

обычно реализуется с помощью операций разбиения строк, таких как `String.split()` в Java, или с помощью простого анализа сканирования, например, `strtok()` в C или C++.

Строка DEF представляет собой серию из одного или нескольких элементов, разделенных точками с запятой (;). Каждый элемент состоит из строки элемента, за которой могут следовать двоеточие (:) и список параметров. Список параметров - это одна или несколько строк параметров, разделенных запятыми (,).

Обратите внимание, что строковые параметры не окружены разделителями, такими как кавычки, однако специальные символы в параметрах строки будут исключены с помощью амперсанда (&), как описано в пункте 13-6.1.2.

13-6.1.1 Специальные символы

В следующей таблице перечислены специальные символы, используемые в DEF.

Таблица 13-1 – Специальные символы

Специальные символы	Использование
Semicolon (;)	Разделяет отдельные элементы DEF.
Colon (:)	Разделяет каждый элемент на строку элемента и список параметров.
Comma (,)	Разделяет отдельные параметры списка параметров.
Ampersand (&)	Исключает / кодирует специальные символы, содержащиеся в DEF.

13-6.1.2 Кодирование строк

Специальные символы, содержащиеся в DEF, удаляются / кодируются с использованием последовательностей символов, перечисленных в следующей таблице.

Таблица 13-2 – Кодирование строк

Специальные символы	Кодирование
Semicolon (;)	&s
Colon (:)	&c
Comma (,)	&m
Ampersand (&)	&a

Например, условное DEF, содержащее четыре элемента, которые могут использоваться для представления инструкций по отрисовке:

PenWidth:0.64;PenColor:LANDF,0.75;DrawLine;DrawTextStrings:Hello&m world!,,Foo&cbar

Первый элемент имеет один параметр (0.64), второй элемент имеет два параметра (LANDF и 0.75), третий элемент не имеет параметров, а четвертый элемент имеет три параметра (Hello, world!, null или empty и Foo:bar).

13-6.1.3 Парсинг

Есть четыре шага для парсинга DEF: (1) получить каждый элемент, (2) получить элемент и параметры для каждого элемента, (3) разбить параметры на отдельные части, а затем (4) декодировать каждый параметр. Условный DEF:

Item1:P1A;Item2:P2A,P2B;Item3:Hello&m world!

Сначала хост должен разделить DEF на отдельные элементы на каждой границе точки с запятой (;), что приведет к следующему:

Таблица 13-3 – Парсинг – Шаг 1

ELEMENT #	ELEMENT
1	<i>Item1:P1A</i>
2	<i>tem2:P2A,P2B</i>
3	<i>Item3:Hello&m world!</i>

Затем каждый из элементов следует разделить на элемент и параметр(ы) элементов, разделив границы двоеточием (:), в результате чего:

Таблица 13-4 – Парсинг – Шаг 2

ELEMENT #	ELEMENT	ITEM	PARAMETERS
1	<i>Item1:P1A</i>	<i>Item1</i>	<i>P1A</i>
2	<i>tem2:P2A,P2B</i>	<i>Item2</i>	<i>P2A,P2B</i>
3	<i>Item3:Hello&m world!</i>	<i>Item3</i>	<i>Hello&m world!</i>

Затем параметры должны быть индивидуально извлечены путем разделения параметров на каждой границе запятой (,), в результате чего:

Таблица 13-5 – Парсинг – Шаг 3

ELEMENT #	ELEMENT	ITEM	PARAMETER 1	PARAMETER 2	...	PARAMETER N
1	<i>Item1:P1A</i>	<i>Item1</i>	<i>P1A</i>			
2	<i>tem2:P2A,P2B</i>	<i>Item2</i>	<i>P2A,P2B</i>	<i>P2B</i>		
3	<i>Item3:Hello&m world!</i>	<i>Item3</i>	<i>Hello&m world!</i>			

После того, как DEF разделен на составные части, каждый параметр должен быть преобразован в его первоначальную кодировку строк путем выполнения замен, перечисленных в таблице 13-6:

Таблица 13-6 – Парсинг – Шаг 4

ELEMENT #	ELEMENT	ITEM	PARAMETER 1	PARAMETER 2	...	PARAMETER N
1	<i>Item1:P1A</i>	<i>Item1</i>	<i>P1A</i>			
2	<i>tem2:P2A,P2B</i>	<i>Item2</i>	<i>P2A</i>	<i>P2B</i>		
3	<i>Item3:Hello&mworld!</i>	<i>Item3</i>	<i>Hello, world!</i>			

13-6.2 Путь к атрибуту

Каталоги скриптинга должны иметь возможность определять значение атрибутов для каждой реализации фичера, содержащегося в наборе данных. Для этого каталог запросит хост для каждого значения атрибута по мере необходимости. При запросе хоста каталог должен определить, какой атрибут данного фичера запрашивается. Если реализация фичера содержит только простые атрибуты, определения реализации фичера и кода атрибута достаточно, чтобы хост однозначно идентифицировал запрошенный атрибут.

Хосту требуется больше информации, когда значение атрибута содержится в сложном атрибуте. Например, рассмотрим следующий поиск значения атрибута:

```
feature.sectorCharacteristic[2].lightSector[1].valueOfNominalRange
```

Здесь фичер имеет сложный атрибут *sectorCharacteristic*, который представляет собой массив. Вторая запись *sectorCharacteristic* содержит сложный атрибут *lightSector*, первый элемент которого содержит простой атрибут *valueOfNominalRange*.

Когда запрашиваем значение *valueOfNominalRange*, скриптинг должен предоставить хосту путь к желаемому атрибуту, в дополнение к коду желаемого атрибута, чтобы хост мог передать фактическое значение. Путь требуется, поскольку реализация фичера может иметь несколько реализаций атрибута с одним и тем же кодом, содержащимся в альтернативных путях к атрибуту - например:

```
feature.simpleAttribute, vs. feature.complexAttribute[n].simpleAttribute vs.  
feature.complexAttribute[n+1].simpleAttribute.
```

Когда каталог скриптинга запрашивает у хоста значение атрибута, путь к атрибуту предоставляется хосту с помощью строки DEF. Каждая секция пути кодируется как элемент, содержащий *AttributeCode* и *Index*. *AttributeCode* содержит код комплексного атрибута; *Index* сохраняет индекс массива комплексного атрибута.

В приведенном выше примере путь к *valueOfNominalRange* будет выражен в DEF следующим образом:

```
sectorCharacteristic:2;lightSector:1
```

DEF будет использоваться для вызова хоста из каталога скриптинга следующим образом:

```
HostFeatureGetSimpleAttribute(featureID,  
sectorCharacteristic:2;lightSector:1, valueOfNominalRange)
```

13-7 Требования к хостингу

В этом разделе определяются требования, предъявляемые к хосту для поддержки функциональности сценариев. Например, программа, написанная для отображения ЭНК S-101 с использованием S-100 Часть 9A, должна соответствовать требованиям этого раздела.

13-7.1 Версия Lua

Хост должен предоставлять движок сценариев; интерпретатор Lua версии 5.1 или виртуальную машину. Реализация ссылки доступна на lua.org (<http://www.lua.org/>). Рекомендуется встроить эталонную реализацию в хостинг. Для максимальной производительности хост может встраивать или реализовывать компилятор Lua, такой как LuaJIT (<http://luajit.org/>).

Дополнительные руководящие указания по внедрению смотрите в *Programming in Lua – Part IV (The C API)*, подробности которых смотрите на <https://www.lua.org/pil/>.

13-7.2 Кодирование символов

Все строки, обмениваемые между хостом и каталогом скриптинга, должны кодироваться в UTF-8.

13-7.3 Обработка ошибок

При вызове функций каталога скриптинга Lua с хоста возвращается значение LUA_OK из lua_rcall, указывающее на успех. В противном случае используется стандартный механизм обработки ошибок Lua.

Код ошибки возвращается хосту, и строка с подробным описанием ошибки будет доступна в верхней части стека.

13-7.4 Параметры массивов

Некоторые функции каталога скриптинга предполагают, что массивы будут передаваться в качестве параметров. Массивы являются стандартными массивами Lua, которые должны создаваться с использованием функций массивов Lua C API, как это описано в *Programming in Lua – Part IV (The C API)*.

13-7.5 Функции хостов

Хост должен предоставлять стандартные функции хоста, описанные в пункте 13-8.2.

Хост должен также предоставлять функции для конкретного домена, чтобы поддерживать специфические функции домена. Специальные функции домена, которые не используются хостом, предоставлять не нужно. Документация по функциям хоста для конкретного домена представлена в Части(ях) S-100, описывающей специфическую для домена функциональность.

13-7.5.1 Совместимость

Хост должен гарантировать обратную совместимость функций хоста со всеми ранее опубликованными каталогами скриптинга. То есть, при реализации функции X хост должен вызывать только те функции каталога сценариев, которые были доступны в версии S-100 при добавлении X.

Несоблюдение этого требования может привести к несовместимости, когда хост пытается запустить старые каталоги скриптинга.

13-7.5.1.1 Несовместимость каталога скриптинга / хоста

При публикации новых версий S-100 могут быть добавлены функции скриптинга. Скриптовые функции никогда не будут удалены из S-100, хотя использование конкретной функции может быть отменено.

Хотя обратная совместимость гарантирована, новые каталоги скриптинга могут попытаться вызвать функции хоста, которые не поддерживаются текущим хостом. Эта ситуация указывает на хост, который не был обновлен с последними функциями скриптинга хоста. Для ограничения таких случаев, каталоги скриптинга должны быть написаны с использованием самого раннего подмножества скриптовых функций.

Несоответствия скриптинга (отсутствующие функции хоста) указываются при инициализации скриптинга. Несовместимость указывается на хосте возвращением **LUA_ERRERR** из lua_pcall; строка ошибки в верхней части стека будет подробно описывать причину несовместимости. Когда это происходит, хост должен вернуться к более ранней версии каталога скриптинга, если она доступна. Также рекомендуется предупредить пользователя, чтобы он проверил обновление программного обеспечения хоста.

13-8 Функции стандартных скриптов

В этом разделе описывается набор стандартных функций скриптов, которые составляют систему скриптинга. Описаны два набора функций: стандартные функции каталога и стандартные функции хоста. Реализация каталога скриптинга существует только в домене скриптинга.

Стандартные функции каталога, описанные в пункте 13-8.1, предусмотрены в каждом каталоге скриптинга. Стандартные функции хоста, описанные в пункте 13-8.2, должны быть реализованы программой, в которой находится среда скриптинга.

На рисунке 13-3 ниже показано расположение каждого типа функции скриптинга в среде скриптинга.



Рисунок 13-3 – Расположение функций скриптинга в среде скриптинга

Каждая стандартная функция скриптинга описывается ниже в подпунктах. Приведено описание назначения функции, а также описание параметров и возвращаемого значения.

Для ясности *void* используется для указания того, что функция не имеет возвращаемого значения.

Функциональные параметры, которые могут принимать несколько типов, будут указаны в качестве *variant*. Этот *variant* будет также использоваться, если функция может возвращать более одного типа. Например, функция, которая принимает как целые числа, так и строки для первого параметра и возвращает либо целое число, либо строку, зависящую от типа, переданного для первого параметра, будет иметь подпись:

variant **Function**(*variant param1*)

Описание функции будет указывать типы, разрешенные для варианта параметра(ов).

Многие функции стандартного скрипта принимают параметр *featureID*, *informationTypeID* или *spatialID*. Хост должен убедиться, что эти различные параметры идентификатора однозначно идентифицируют одну реализацию среди всех наборов данных по всем типам продуктов, которые будут использоваться хостом во время сеанса скриптинга. Поскольку каждый тип идентификатора является строкой, одним из способов достижения этого является ввод соответствующей информации в идентификатор; например, "S101.101US003DE01M_.000.F1" для идентификации первого фичера в указанном наборе данных S-101.

13-8.1 Стандартные функции каталога

В этом разделе описывается стандартный набор функций, предоставляемых всеми каталогами скриптинга.

Все строки, передаваемые этим функциям, должны быть закодированы в UTF-8.

При вызове этих функций значения атрибутов всегда передаются из хоста в среду скриптинга с помощью строк. Это позволяет однозначно передавать значения, не имеющие эквивалентов Lua. Это также позволяет передавать десятичные значения без потери точности, которая может произойти при преобразовании в типы с плавающей запятой IEEE.

Если значение атрибута присутствует, но неизвестно, следует использовать значение, возвращенное из *GetUnknownAttributeString()*.

В следующей таблице показаны строковые представления типов, определенных *S100_CD_AttributeValueType*.

Таблица 13-7 – Строковое представление типов, определяемых *S100_CD_AttributeValueType*

S100_CD_AttributeValueType	Представление
boolean	"0" представляет False "1" представляет True
enumeration	S100_FC_ListedValue:code. Не используйте S100_FC_ListValue:label
integer	Строковое представление целого числа со знаком
real	Строковое представление десятичного числа. Конечные нули допускаются только в том случае, если они значимы.
text	Как это предусмотрено
date	Кодировка должна соответствовать дате по ISO 8601
time	Кодировка должна соответствовать формату времени, указанному в ISO 8601
dateTime	Кодировка должна соответствовать формату даты и времени, указанному в ISO 8601
URI	Кодировка должна соответствовать формату URI, указанному в RFC 3986
URL	Кодировка должна соответствовать формату URL, указанному в RFC 3986
URN	Кодировка должна соответствовать формату URN, определенному в RFC 2141
S100_CodeList	Как это предусмотрено
S100_TruncatedDate	Как это предусмотрено

13-8.1.1 Функции создания объектов

Эти функции освобождают хост от бремени построения таблиц Lua, соответствующих комплексным типам, используемым в каталоге скриптинга. Они позволяют хосту создавать объекты, которые будут переданы в каталог скриптинга. Схема и содержимое созданных объектов непрозрачны для хоста - они предназначены только для использования в каталоге скриптинга.

13-8.1.1.1 AttributeConstraints CreateAttributeConstraints(integer *stringLength*, string *textPattern*, string *rangeLower*, string *rangeUpper*, string *rangeClosure*, integer *precision*)

Возвращаемое значение:

AttributeConstraints

Таблица Lua, содержащая объект ограничений атрибутов.

Параметры:

stringLength: integer или ноль

Максимальное число символов, которое может быть назначено типу текстового атрибута. Если это значение равно нулю, длина не ограничена.

textPattern: string или ноль

Регулярное выражение, определяющее структуру текстовых значений, которые могут быть назначены атрибуту. Если это значение равно нулю, то структура является неограниченной.

W3C XML Standard Part 2 Appendix F (Регулярные выражения) используются для определения шаблона текста.

rangeLower: string или ноль

Указывает нижний диапазон допустимых значений атрибута. Если это значение равно нулю, то не существует нижнего ограничения.

rangeUpper: string или ноль

Задаёт верхний диапазон допустимых значений атрибута. Если это значение равно нулю, верхнего ограничения не существует.

rangeClosure: string или ноль

Определяет операции закрытия для нижнего и верхнего диапазонов. Это одно из перечисленных значений, определенных в таблице 1-3. Это должно быть определено, если указан либо нижний, либо верхний диапазон.

precision: integer или ноль

Если указано, определяет точность вещественного числа.

Примечания:

Вызывается из хоста для создания ограничений атрибутов для использования в каталоге скриптинга. Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.2 SpatialAssociation CreateSpatialAssociation (string *spatialType*, string *spatialID*, string *orientation*, variant *scaleMinimum*, variant *scaleMaximum*)

Возвращаемое значение:

SpatialAssociation

Таблица Lua, содержащая объект пространственной ассоциации.

Параметры:

spatialType: строка

Тип пространственный. Один из: "Point", "MultiPoint", "Curve", "CompositeCurve" или "Surface".

spatialID: строка

Используется хостом для уникальной идентификации пространственного типа.

orientation: строка

Ориентация пространственного типа. Один из Forward или Reverse.

scaleMinimum: integer или ноль

Минимальный масштаб отображения для пространственного типа или ноль.

scaleMaximum: integer или ноль

Максимальный масштаб отображения для пространственного типа или ноль.

Примечания:

Вызывается с хоста для создания пространственной ассоциации для использования в каталоге скриптинга.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.3 Point CreatePoint(string *x*, string *y*, variant *z*)

Возвращаемое значение:

Point

Таблица Lua, содержащая точечный объект.

Параметры:

x: строка

X координата точки.

y: строка

Y координата точки.

Z: строка или нуль

Z координата точки. Для 2D точек это значение должно быть равно *nil*.

Примечания:

x, y и z выражаются с использованием строкового представления *real*, описанного в пункте 13-8.1.

Вызывается из хоста для создания точечного пространственного объекта для использования в каталоге скриптинга.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.4 MultiPoint CreateMultiPoint(Point[] *points*)

Возвращаемое значение:

MultiPoint

Таблица Lua, содержащая многоточечный объект.

Параметры:

points: Point[]

Массив точек Lua. Хост создает каждую точку вызывая *CreatePoint*.

Примечания:

Вызывается из хоста для создания многоточечного пространственного объекта для использования в каталоге сценариев. Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог сценариев.

13-8.1.1.5 CurveSegment CreateCurveSegment(Point[] *controlPoints*, string *interpolation*)

Возвращаемое значение:

CurveSegment

Таблица Lua, содержащая объект сегмента кривой.

Параметры:

controlPoints: Point[]

Массив точек, определяющий контрольные точки сегмента кривой. Хост создает каждую *controlPoint*, вызывая *CreatePoint*.

Interpolation: string

Интерполяция, используемая при соединении контрольных точек. Один из

S100_CurveInterpolationL:name.

Примечания:

Вызывается из хоста для создания пространственного объекта сегмента кривой.

Не предполагается, что узел будет манипулировать возвращенным объектом; объект предназначен для передачи от узла обратно в каталог скриптинга.

13-8.1.1.6 Curve CreateCurve(Point *startPoint*, Point *endPoint*, CurveSegment[] *segments*)

Возвращаемое значение:

Curve

Таблица Lua, содержащая объект кривая.

Параметры:

startPoint: Point

Начальная точка кривой. Создается хостом, вызывая *CreatePoint*.

endpoint: Point

Конечная точка кривой. Создается хостом, вызывая *CreatePoint*.

segments: CurveSegment[]

Массив сегментов кривой, составляющих кривую. Каждый элемент массива создается вызовом *CreateCurveSegment*.

Примечания

Вызывается из хоста для создания пространственного объекта кривой.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.7 CompositeCurve CreateCompositeCurve(SpatialAssociation[] *curveAssociations*)

Возвращаемое значение:

CompositeCurve

Таблица Lua, содержащая объект составной кривой.

Параметры:

curveAssociations: SpatialAssociation[]

Массив пространственных ассоциаций, определяющих элементы композитной кривой. Хост создает каждый *SpatialAssociation*, вызывая *CreateSpatialAssociation*.

Примечания:

Вызывается из хоста для создания пространственного объекта составной кривой.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.8 Поверхность `CreateSurface(SpatialAssociation exteriorRing, вариант interiorRings)`

Возвращаемое значение:

Surface

Таблица Lua, содержащая объект поверхность.

Параметры:

exteriorRing: *SpatialAssociation*

Пространственная ассоциация кольца, определяющая внешнее кольцо поверхности. Создается хостом, вызывая *CreateSpatialAssociation*.

interiorRings: *SpatialAssociation[]* или нуль

Определяет "дырки" внутри поверхности. Хост создает каждое внутреннее кольцо, вызывая *CreateSpatialAssociation*. Если нет отверстий, этот параметр равен нулю.

Примечания:

Вызывается от хоста для создания пространственного объекта поверхность.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.9 `ArcByCenterPoint CreateArcByCenterPoint(SpatialAssociation centerPoint, real radius, real startAngle, real angularDistance)`

Возвращаемое значение:

ArcByCenterPoint

Таблица Lua, содержащая объект *ArcByCenterPoint*.

Параметры:

centerPoint: *SpatialAssociation*

Пространственная ассоциация точки, определяющая центральную точку дуги. Хост создает, вызывая *CreateSpatialAssociation*.

radius: *real*

Определяет геодезическое расстояние от центра.

startAngle: *real*

Начальный пеленг дуги в градусах, диапазон ограничен [0.0, 360.0].

angularDistance: real

Угловое расстояние дуги в градусах, диапазон ограничен [-360.0, 360.0]. Положительные числа указывают направление по часовой стрелке.

Примечания:

Вызывается с хоста для создания пространственного объекта *ArcByCenterPoint*. Дуга начинается с пеленга, заданного параметром *startAngle*, и заканчивается на пеленге, рассчитанном путем добавления значения параметра *angularDistance* к начальному углу. Направление дуги задается знаком углового расстояния. Пеленги проводятся относительно истинного севера, за исключением дуг на обоих полюсах (где истинный север не определен или неоднозначен), где они должны использовать главный меридиан в качестве начального направления.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.10 CircleByCenterPoint CreateCircleByCenterPoint(SpatialAssociation centerPoint, real radius, real startAngle, real angularDistance)

Возвращаемое значение:

CircleByCenterPoint

Таблица Lua, содержащая объект *CircleByCenterPoint*.

Параметры:

centerPoint: SpatialAssociation

Пространственная ассоциация точки, которая определяет центральную точку окружности. Хост создает, вызывая *CreateSpatialAssociation*.

radius: real

Определяет геодезическое расстояние от центра.

startAngle: real

Опциональный. Начальный пеленг дуги в градусах, диапазон ограничен [0,0, 360,0]. По умолчанию равен нулю.

angularDistance: real

Опциональный. Угловое расстояние окружности в градусах должно быть либо -360,0 (против часовой стрелки), либо 360,0 (по часовой стрелке). Положительные числа указывают направление по часовой стрелке. По умолчанию - 360 (по часовой стрелке).

Примечания:

Вызывается с хоста для создания объекта *CircleByCenterPoint*.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.1.11 SplineCurve CreateSplineCurve(Point[] *controlPoints*, string *interpolation*, integer *degree*, Knot[] *knots*, KnotType *knotSpec*, boolean *isRational*)

Возвращаемое значение:

SplineCurve

Таблица Lua, содержащая сплайн.

Параметры:

controlPoints: Point[]

Массив точек, определяющих контрольные точки сегмента кривой. Хост создает каждый *controlPoint*, вызывая *CreatePoint*. Число контрольных точек должно быть не менее трех.

interpolation: string

Интерполяция, используемая при соединении контрольных точек. Одна из S100_CurveInterpolation:name.

degree: integer

Степень многочленов, используемых для определения интерполяции.

knots: Knot[]

Массив узлов. Каждый узел определяет параметр в пространстве параметров сплайна, который используется для определения базисной функции сплайна. Каждый узел создается путем вызова *CreateKnot*.

knotSpec: KnotType

Тип распределения узлов при определении сплайна. Определяется S100_GM_KnotType.

isRational: boolean

Указывает, использует ли сплайн рациональные функции для определения кривой.

Примечания:

Вызывается из хоста для создания сплайн-кривой пространственного объекта.

Не предполагается, что узел будет манипулировать возвращенным объектом; объект предназначен для передачи из хоста обратно в каталог скриптинга.

13-8.1.1.12 PolynomialSpline CreatePolynomialSpline(Point[] controlPoints, string interpolation, integer degree, Knot[] knots, KnotType knotSpec, Vector[] derivativeAtStart, Vector[] derivativeAtEnd, integer numDerivativeInterior)

Возвращаемое значение:

PolynomialSpline

Таблица Lua, содержащая полиномиальный сплайн.

Параметры:

controlPoints: Point[]

Массив точек, определяющих контрольные точки сегмента кривой. Хост создает каждый controlPoint, вызывая *CreatePoint*. Число контрольных точек должно быть не менее трех.

interpolation: string

Интерполяция, используемая при соединении контрольных точек. Одна из S100_CurveInterpolation:name.

degree: integer

Степень многочленов, используемых для определения интерполяции.

knots: Knot[]

Массив узлов. Каждый узел определяет параметр в пространстве параметров сплайна, который используется для определения базисной функции сплайна. Каждый узел создается путем вызова *CreateKnot*.

knotSpec: KnotType

Тип распределения узлов при определении сплайна. Определяется в S100_GM_KnotType.

derivativeAtStart: Vector[]

Массив вектора, который определяет значения начальной производной, используемой для интерполяции на этой кривой в начальной точке сплайна. Для *degree* - 2 вектора могут быть определены. Каждый вектор создается путем вызова *CreateVector*.

derivativeAtEnd: Vector[]

Массив вектора, который определяет значения конечной производной, используемой для интерполяции на этой кривой в конечной точке сплайна. Для *degree* - 2 вектора могут быть определены. Каждый вектор создается путем вызова *CreateVector*.

numDerivativeInterior: KnotType

Количество непрерывных производных, необходимых для внутренних узлов.

Примечания:

Вызывается из хоста для создания полиномиального сплайна пространственного объекта.

Не предполагается, что узел будет манипулировать возвращенным объектом; объект предназначен для передачи из хоста обратно в каталог скриптинга.

13-8.1.1.13 Knot CreateKnot(string *value*[, integer *multiplicity*])

Возвращаемое значение:

Knot

Таблица Lua, содержащая объект узел.

Параметры:

value: string

Значение узла.

multiplicity: integer

Множественность узла. Если не указано, кратность равна единице.

Примечания:

Значения *value* выражаются с помощью представления вещественной строки, как это описано в пункте 13-8.1.

Вызывается из хоста для создания объекта узел.

Не предполагается, что узел будет манипулировать возвращенным объектом; объект предназначен для передачи из хоста обратно в каталог скриптинга.

13-8.1.1.14 Vector CreateVector(Point *origin*, string[] *offset*, integer *dimension*, string *coordinateSystem*)

Возвращаемое значение:

Knot

Таблица Lua, содержащая объект узел.

Параметры:

origin: Point

Расположение точки GeometricReferenceSurface, к которой вектор является касательным.

offset: string[]

Локальный касательный вектор в терминах дифференциалов локальных координат. Значения смещения - величина вектора вдоль каждой оси координат.

dimension: integer

Размеры источника.

coordinateSystem: string

Система координат источника (например EPSG:4326).

Примечания:

Значения *offset* выражаются с помощью представления вещественной строки, как это описано в пункте 13-8.1.

Вызывается из хоста для создания векторного объекта.

Не предполагается, что узел будет манипулировать возвращенным объектом; объект предназначен для передачи из хоста обратно в каталог скриптинга.

13-8.1.2 Функции создания типа информации

Эти функции освобождают хост от бремени построения таблиц Lua, соответствующих комплексным типам, используемым в каталоге скриптинга. Они позволяют хосту создавать объекты, используемые при вызове в каталог скриптинга. Схема и содержимое созданных объектов непрозрачны для хоста - они предназначены только для использования в каталоге скриптинга.

Комплексные типы соответствуют классам, описанным в S-100 Часть 5 - *Каталог фичеров*. Каждая функция создания типа информации, описанная в этом разделе, указывает на соответствующий тип S-100 Часть 5 Каталог фичеров.

Функции создания *FC_DefinitionReference* и их зависимые типы, включая класс *CI_Citation* намеренно пропущены. Случаев идентифицированного использования *FC_DefinitionReference* нет, а применение *CI_Citation* было бы более сложным, чем полный текст этого раздела в его нынешнем виде.

13-8.1.2.1 Item CreateItem(string code, string name, string definition, string remarks, string[] alias)

Возвращаемое значение:

Item

Таблица Lua, содержащая элемент, соответствующий *S100_FC_Item*.

Параметры:

code: string

Код, который однозначно идентифицирует именованный тип в каталоге фичеров.

name: string

Имя элемента.

definition: string

Определение именованного типа на естественном языке.

remarks: string

Дополнительно. Дополнительные разъяснения по данному элементу.

alias: string[]

Эквивалентное наименование(а) данного элемента.

Примечания:

Вызывается с хоста для создания элемента.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.2 NamedType CreateNamedType(Item *item*, boolean *abstract*, AttributeBinding[] *attributeBindings*)

Возвращаемое значение:

NamedType

Таблица Lua, содержащая именованный тип, соответствующий *S100_FC_NamedType*.

Параметры:

item: Item

Реализация элемента, созданная вызовом *CreateItem()*.

abstract: boolean

Указывает, могут ли реализации этого именованного типа существовать в географическом наборе данных. Абстрактные типы не могут быть реализациями, но служат базовыми классами для других (не абстрактных) типов.

attributeBindings: AttributeBinding[]

Массив с нулевыми или более привязками к атрибутам, описывающим характеристику этого именованного типа.

Примечания:

Вызывается с хоста для создания именованного типа.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.3 **ObjectType CreateObjectType(NamedType *namedType*, InformationBinding[] *informationBindings*)**

Возвращаемое значение:

ObjectType

Таблица Lua, содержащая тип объекта, соответствующий *S100_FC_ObjectType*.

Параметры:

namedType: NamedType

Реализация именованного типа, созданного вызовом *CreateNamedType()*.

informationBindings: InformationBinding[]

Массив с нулевыми или более привязками к типам информации, которые могут быть ассоциированы с данным типом объектов с помощью ассоциации информации.

Примечания:

Вызывается с хоста для создания типа объекта.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.4 **InformationType CreateInformationType(ObjectType *objectType*, string *superType*, string[] *subType*)**

Возвращаемое значение:

InformationType

Таблица Lua, содержащая тип информации, соответствующий *S100_FC_InformationType*.

Параметры:

objectType: ObjectType

Реализация поименованного типа, созданная вызовом *CreateObjectType()*.

superType: string

Оptionальный. Указывает тип информации, из которого получен этот тип.

subtype: string[]

Массив из нуля или более типов информации, полученных от этого типа.

Примечания:

Вызывается из хоста для создания типа информации.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.5 FeatureType CreateFeatureType(ObjectType *objectType*, string *featureUseType*, string[] *permittedPrimitives*, FeatureBinding[] *featureBindings*, string *superType*, string [] *subType*)

Возвращаемое значение:

FeatureType

Таблица Lua, содержащая тип фичеров, соответствующий *S100_FC_FeatureType*.

Параметры:

objectType: ObjectType

Реализация именованного типа, созданная вызовом *CreateObjectType()*.

featureUseType: string

S100_CD_FeatureUseType:Name.

permittedPrimitives: string[]

Массив, содержащий ноль или более допустимых пространственных типов примитивов для типа фичеров. Каждый элемент является *S100_FC_SpatialPrimitiveType:Name*.

featureBindings: FeatureBinding[]

Массив нулевых или более привязок к типам фичеров, которые могут быть связаны с этим типом фичера с помощью ассоциации фичеров.

superType: string

Опциональный. Указывает тип фичера, от которого он получен. Подтип будет наследовать все свойства из его супертипа: Имя, определение и код обычно переписываются подтипом, хотя к подтипу могут добавляться новые свойства.

subType: string[]

Массив из нуля или более типов фичеров, производных от этого типа.

Примечания:

Вызывается из хоста для создания типа фичера.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.6 InformationAssociation CreateInformationAssociation(NamedType *namedType*, Role[] *roles*, string *superType*, string[] *subType*)

Возвращаемое значение:

InformationAssociation

Таблица Lua, содержащая ассоциацию информации, соответствующей S100_FC_InformationAssociation.

Параметры:

namedType: NamedType

Элемент поименованного типа, созданный вызовом *CreateNamedType()*.

roles: Role[]

Массив от 0 до 2 ролей ассоциации.

superType: string

Опциональный. Указывает ассоциацию информации, из которой получается эта ассоциация.

subType: string[]

Массив нулевых или более ассоциаций информации, полученных из этой ассоциации.

Примечания:

Вызывается из хоста для создания ассоциации информации.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.7 FeatureAssociation CreateFeatureAssociation(NamedType *namedType*, Role[] *roles*, string *superType*, string [] *subType*)

Возвращаемое значение:

FeatureAssociation

Таблица Lua, содержащая ассоциацию фичера, соответствующую S100_FC_FeatureAssociation.

Параметры:

namedType: NamedType

Элемент поименованного типа, созданный вызовом *CreateNamedType()*.

roles: Role[]

Массив от 0 до 2 ролей ассоциации.

superType: string

Опциональный. Указывает ассоциацию фичеров, из которой получена эта ассоциация.

subType: string[]

Массив из нулевых или более ассоциаций фичеров, полученных из этой ассоциации.

Примечания:

Вызывается из хоста для создания ассоциации фичеров.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.8 Role CreateRole(Item *item*)

Возвращаемое значение:

Role

Таблица Lua, содержащая роль, соответствующую *S100_FC_Role*.

Параметры:

item: Item

Реализация элемента, созданного вызовом *CreateItem()*.

Примечания:

Вызывается из хоста для создания роли.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.9 SimpleAttribute CreateSimpleAttribute(Item *item*, string *valueType*, string *uom*, string *quantitySpecification*, AttributeConstraints *attributeConstraints*, ListedValue[] *listedValues*)

Возвращаемое значение:

SimpleAttribute

Таблица Lua, содержащая простой атрибут, соответствующий *S100_FC_SimpleAttribute*.

Параметры:

item: string

Реализация элемента, созданного вызовом *CreateItem()*.

valueType: string

Тип значения атрибута этого фичера. *S100_CD_AttributeValueType:Name*.

uom: string

Опциональный. Единица измерения, используемая для значений атрибута фичера. *S100_UnitOfMeasure:Name*.

quantitySpecification: string

Опциональный. Спецификация количества.
S100_CD_QuantitySpecification:Name.

attributeConstraints: AttributeConstraints

Опциональный. Ограничения, которые могут применяться к атрибуту. Создается вызовом *CreateAttributeConstraints()*.

listedValues: ListedValue[]

Массив нулевых или более списочных значений для домена перечисляемых атрибутов. Каждое списочное значение создается вызовом *CreateListedValue()*. Применяется только, если *valueType* типа *Enumeration* или *S100_Codelist* (с *codelistType* открытый перечень).

Примечания:

Вызывается из хоста для создания типа простого атрибута объекта информации.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.10 ComplexAttribute CreateComplexAttribute(Item item, AttributeBinding[] subAttributeBindings)

Возвращаемое значение:

ComplexAttribute

Таблица Lua, содержащая комплексный атрибут, соответствующий *S100_FC_ComplexAttribute*.

Параметры:

tem: Item

Реализация элемента, созданного вызовом *CreateItem()*.

subAttributeBindings: AttributeBinding[]

Массив из одного или нескольких привязок атрибутов к податрибутам.

Примечания:

Вызывается из хоста для создания комплексного типа атрибута объекта информации.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.11 ListedValue CreateListedValue(string *label*, string *definition*, integer *code*, string *remarks*, string[] *aliases*)

Возвращаемое значение:

ListedValue

Таблица Lua, содержащая списочное значение, соответствующее *S100_FC_ListedValue*.

Параметры:

label: string

Описательная метка, которая однозначно определяет одно значение атрибута фичера.

definition: string

Определение списочных значений на естественном языке.

code: integer

Цифровой код, который однозначно идентифицирует списочное значение для соответствующего атрибута фичера. Положительное целое число.

remarks: string

Опциональный. Дополнительное разъяснение относительно списочного значения.

aliases: string[]

Опциональный. Массив с нулевым или более эквивалентным именем (именами) этого списочного значения.

Примечания:

Вызывается из хоста для создания типа списка значений объекта информации.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.12 AttributeBinding CreateAttributeBinding(string *attributeCode*, integer *lowerMultiplicity*, integer *upperMultiplicity*, boolean *sequential*, integer[] *permittedValues*)

Возвращаемое значение:

AttributeBinding

Таблица Lua, содержащая привязку атрибута, соответствующего *S100_FC_AttributeBinding*.

Параметры:

attributeCode: string

Код комплексного или простого атрибута, связанного с элементом или комплексным атрибутом.

lowerMultiplicity: integer

Минимальное число необходимых повторений этого атрибута. Это ноль для опциональных атрибутов.

upperMultiplicity: integer

Максимальное число допустимых повторений этого атрибута. Равно нулю для бесконечного числа допустимых атрибутов.

sequential: boolean

Описывает, является ли последовательность атрибутов значимой или нет. Применяется только к атрибутам, которые могут возникать более одного раза.

permittedValues: integer[]

Массив нулевых или более допустимых значений атрибута. Каждый ввод - это *S100_FC_ListedValue:code*. Применяется только к атрибутам перечисляемых типов данных.

Примечания:

Вызывается из хоста для создания объекта привязки атрибутов.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.13 InformationBinding CreateInformationBinding(string informationTypeCode, integer lowerMultiplicity, integer upperMultiplicity, string roleType, string role, string association)

Возвращаемое значение:

InformationBinding

Таблица Lua, содержащая привязку информации, соответствующей *S100_FC_InformationBinding*.

Параметры:

informationTypeCode: string

S100_FC_InformationType:code целевого типа информации.

lowerMultiplicity: integer

Минимальное число обязательных значений этого атрибута. Равно нулю для опциональных атрибутов.

upperMultiplicity: integer

Максимальное число допустимых повторений этого атрибута. Равно нулю для бесконечного числа допустимых атрибутов.

roleType: string

Характер конца ассоциации. *S100_FC_RoleType:Name*.

role: string

Опциональный. Код роли, используемой для связывания. Он должен быть частью ассоциации, используемой для связывания, и определяет конец ассоциации.

association: string

Код ассоциации информации, используемый для связывания; определяющий также роль.

Примечания:

Вызывается из хоста для создания объекта привязки информации.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.2.14 FeatureBinding CreateFeatureBinding(string *featureTypeCode*, integer *lowerMultiplicity*, integer *upperMultiplicity*, string *roleType*, string *role*, string *association*)

Возвращаемое значение:

FeatureBinding

Таблица Lua, содержащая связку фичера, соответствующую *S100_FC_FeatureBinding*.

Параметры:

featureTypeCode: string

Код целевого типа фичера.

lowerMultiplicity: integer

Минимальное число обязательных повторений этого атрибута. Для необязательных атрибутов - равно нулю.

upperMultiplicity: integer

Максимальное число допустимых повторений этого атрибута. Равно нулю для бесконечного числа допустимых атрибутов.

roleType: string

Характер конца этой ассоциации. А *S100_FC_RoleType:Name*.

role: string

Код роли, используемый для связывания. Он должен быть частью ассоциации, используемой для связывания, и определяет конец ассоциации.

association: string

Код ассоциации фичеров, используемый для связывания.

Примечания:

Вызывается из хоста для создания объекта привязки фичеров.

Не предполагается, что хост будет манипулировать возвращенным объектом; объект предназначен для передачи от хоста обратно в каталог скриптинга.

13-8.1.3 Другие функции

Функции, описанные на следующих страницах, не подпадают под одну из описанных ранее функций.

13-8.1.3.1 string GetUnknownAttributeString()

Возвращаемое значение:

string

Строка, представляющая значение атрибута, которое присутствует, но неизвестно.

Примечания:

Предназначен для дифференциации неизвестных значений строки от пустых значений строки. Эта функция возвращает постоянное значение.

13-8.1.3.2 string EncodeDEFString(string *input*)

Возвращаемое значение:

string

Кодирование *input*, как описано в пункте 13-6.1.2.

Параметры:

input: string

Кодированная строка.

Примечания:

Кодирует входную строку, как описано в разделе 13-6.1.2.

13-8.1.3.3 string DecodeDEFString(string *encodedString*)

Возвращаемое значение:

string

Раскодированная версия *encodedString*.

Параметры:

encodedString: string

Закодированная строка.

Примечания:

Декодирует входную строку, которая ранее была закодирована, как описано в разделе 13-6.1.2.

13-8.1.3.4 void TypeSystemChecks(boolean *enabled*)

Возвращаемое значение:

Нет

Параметры:

enabled: boolean

Включает или отключает проверку типа.

Примечания:

Указывает тип данных параметра, который должен проверяться при каждом вызове функции. По умолчанию отключен. Проверка системы типов может быть включена во время разработки каталога в качестве средства отладки.

13-8.2 Стандартные функции хоста

Хост должен предоставить набор функций "обратного вызова", которые обеспечивают среду скриптинга: Доступом к реализации хостом общей модели фичеров S-100; доступом к типу информации для любого объекта, определенной моделью; и доступом к пространственным операциям, которые могут использоваться для проведения реляционных испытаний и операций с пространственными элементами, определенной моделью. Хост может дополнительно предоставлять функцию обратного вызова, используемую для взаимодействия с отладчиком.

Разгрузка этих задач для хоста, а не предоставление жестких структур данных, которые передаются между хостом и скриптингом, позволяет хосту взаимодействовать со скриптами, используя оптимальное представление хостом

Общей модели фичеров. Перенос внутренней модели данных хоста на конкретную входную схему не требуется при использовании скриптинга.

Любая из стандартных функций хоста может быть вызвана из каталога скриптинга во время выполнения скрипта.

13-8.2.1 Функции доступа к данным

Хост должен реализовывать функции, описанные на следующих страницах, чтобы позволить среде скриптинга получать доступ к данным, загруженным хостом из набора данных. Эти функции обеспечивают среду скриптинга доступом к фичерам, пространственным данным, значениям атрибутов и информационным ассоциациям.

13-8.2.1.1 `string[] HostGetFeatureIDs()`

Возвращаемое значение:

string[]

Массив Lua, содержащий все ID фичеров в наборе данных.

Примечания:

Инструктирует хост как вернуть все ID фичеров, относящиеся к текущей операции каталога скриптинга.

Это, как правило, все фичеры в *S100_Dataset* или *S100_DataCoverage*.

Как обсуждалось в пункте 13-8, хост отвечает за то, чтобы каждый ID фичеров однозначно идентифицировал одну реализацию фичера среди всех типов продуктов и наборов данных, которые будут использоваться в ходе текущей сессии скриптинга.

13-8.2.1.2 `string HostFeatureGetCode(string featureID)`

Возвращаемое значение:

string

Код, определенный каталогом фичеров для типа реализации фичера.

Параметры:

featureID: string

Используется узлом для уникальной идентификации реализации фичера.

Примечания:

Поручает узлу вернуть код типа фичера для реализации фичера, идентифицированной *featureID*.

13-8.2.1.3 `string[] HostGetInformationTypeIDs()`

Возвращаемое значение:

string[]

Массив Lua, содержащий все ID типов информации в наборе данных.

Примечания:

Позволяет скриптам запрашивать у хоста список типов информации, содержащихся в данном наборе данных.

Инструктирует хост как вернуть массив, содержащий все идентификаторы информации в данном наборе данных.

13-8.2.1.4 `string HostInformationTypeGetCode(string informationTypeID)`

Возвращаемое значение:

string

Код, определенный каталогом фичеров для типа информации реализации информационного типа.

Параметры:

informationTypeID: *string*

Используется хостом для уникальной идентификации реализации типа информации.

Примечания:

Инструктирует хост как вернуть код типа информации для реализации типа информации, указанного в *informationTypeID*.

13-8.2.1.5 `string[] HostFeatureGetSimpleAttribute(string featureID, path path, string attributeCode)`

Возвращаемое значение:

string[]

Текстовое представление каждого значения атрибута, описанное в пункте 13-8.1. Массив возвращается даже если атрибут имеет одно значение.

Параметры:

featureID: *string*

Используется узлом для уникальной идентификации реализации фичера.

path: *path*

Путь к атрибуту, описанный в пункте 13-6.2

attributeCode: string

Один из кодов атрибутов, определенных в каталоге фичеров для типа фичера, определенного в *featureID*.

Примечания:

Инструктирует хост как выполнить простой поиск атрибута по коду *attributeCode* по пути *path* к реализации фичера, идентифицируемой атрибутом *featureID*. Пустой массив возвращается в случае, если запрошенный атрибут отсутствует.

13-8.2.1.6 integer HostFeatureGetComplexAttributeCount(string *featureID*, path *path*, string *attributeCode*)

Возвращаемое значение:

integer

Число соответствующих комплексных атрибутов, существующих по пути к реализации фичера.

Параметры:

featureID: string

Используется хостом для уникальной идентификации реализации фичера.

path: path

Путь к атрибуту, описанный в пункте 13-6.2.

attributeCode: string

Один из кодов атрибутов, определенных в каталоге фичеров для типа фичера, определенного в *featureID*.

Примечания:

Предписывает хосту возвращать число атрибутов, соответствующих *attributeCode*, по заданному пути к атрибуту для данной реализации фичера. Данный путь всегда будет действительным для реализации фичера. Возвращаемое целое число может быть нулём.

13-8.2.1.7 SpatialAssociation[] HostFeatureGetSpatialAssociations(string *featureID*)

Возвращаемое значение:

SpatialAssociation[]

Массив Lua, содержащий все пространственные ассоциации реализации фичера, представляемого в *featureID*.

Параметры:

featureID: string

Используется хостом для уникальной идентификации реализации фичера.

Примечания:

Инструктирует хост как вернуть массив, содержащий пространственные ассоциации данной реализации фичера. Для каждой пространственной ассоциации, которую содержит фичер, хост вызывает стандартную функцию каталога *CreateSpatialAssociation* для создания объекта *SpatialAssociation*.

Хост должен возвращать пустой массив, если функция не имеет пространственных связей.

13-8.2.1.8 string[] HostFeatureGetAssociatedFeatureIDs(string *featureID*, string *associationCode*, variant *roleCode*)

Возвращаемое значение:

string[]

Массив Lua, содержащий ассоциированные ID фичеров.

Параметры:

featureID: string

Используется хостом для уникальной идентификации реализации фичера.

associationCode: string

Код запрашиваемой ассоциации, определенный в каталоге фичеров.

roleCode: string или nil

Код запрашиваемой роли, определенный в каталоге фичеров. Может быть ноль, если *associationCode* сам по себе достаточен для указания ассоциации или если все роли, определенные *associationCode*, являются желательными.

Примечания:

При вызове хост возвращает массив, содержащий идентификаторы фичеров, ассоциированные с данной реализацией фичера, которые совпадают с *associationCode* и *roleCode*. Если совпадения не найдены, то хост возвращает пустой массив.

roleCode может быть ноль, в этом случае только *associationCode* следует использовать для поиска.

13-8.2.1.9 `string[] HostFeatureGetAssociatedInformationIDs(string featureID, string associationCode, variant roleCode)`

Возвращаемое значение:

`string[]`

Массив Lua, содержащий ассоциированную информацию ID.

Параметры:

`featureID`: string

Используется хостом для уникальной идентификации реализации фичера.

`associationCode`: string

Код запрашиваемой ассоциации, определенный в каталоге фичеров.

`roleCode`: string или nil

Код запрашиваемой роли, определенный каталогом фичеров. Может быть ноль, если `associationCode` сам по себе достаточен для указания ассоциации или если все роли, определенные `associationCode`, являются желательными.

Примечания:

При вызове хост возвращает массив, содержащий информацию идентификаторов фичеров, ассоциированных с данной реализацией фичера, которые совпадают с `associationCode` и `roleCode`. Если совпадения не найдены, то хост возвращает пустой массив.

`roleCode` может быть ноль, в этом случае только `associationCode` следует использовать для поиска.

13-8.2.1.10 `string[] HostInformationGetAssociatedInformationIDs(string informationID, string associationCode, variant roleCode)`

Return Value:

`string[]`

Массив Lua, содержащий ассоциированные идентификаторы информации.

Параметры:

`informationID`: string

Используется узлом для уникальной идентификации реализации информации.

`associationCode`: string

Код запрашиваемой ассоциации, определенный в каталоге фичеров.

`roleCode`: string или nil

Код запрашиваемой роли, определенный каталогом фичеров. Может быть ноль, если *associationCode* сам по себе достаточен, чтобы указать связь или если требуются все роли определенные *associationCode*.

Примечания:

При вызове хост возвращает массив, содержащий идентификаторы информации, ассоциированные с соответствующей реализацией информации *associationCode* и *roleCode*. Если совпадения не найдены, то хост возвращает пустой массив.

roleCode может быть нулем, в этом случае для поиска используется *associationCode*.

13-8.2.1.11 *string[]* HostGetSpatialIDs()

Возвращаемое значение:

string[]

Массив Lua, содержащий все пространственные идентификаторы в наборе данных.

Примечания:

Поручает хосту вернуть все пространственные идентификаторы, относящиеся к текущей операции каталога скриптинга.

Это, как правило, все пространственные объекты в *S100_Dataset* или *S100_DataCoverage*.

Как обсуждалось в пункте 13-8, хост отвечает за то, чтобы каждый пространственный идентификатор однозначно идентифицировал одну пространственную реализацию среди всех типов продуктов и наборов данных, которые будут использоваться в ходе текущей сессии скриптинга.

13-8.2.1.12 Spatial HostGetSpatial(string *spatialID*)

Возвращаемое значение:

Spatial

Пространственный объект, созданный с помощью стандартной функции каталога, как указано в примечаниях.

Параметры:

spatialID: string

Используется хостом для уникальной идентификации пространственных данных.

Примечания:

Запросы хоста для заданных пространственных данных.

Хост возвращает пространственный объект, созданный одной из стандартных функций каталога, определенных в пункте 13-8.1.1.

13-8.2.1.13 variant HostSpatialGetAssociatedInformationIDs(string *spatialID*, string *associationCode*, variant *roleCode*)

Возвращаемое значение:

nil

Ассоциация информации не действительная для данного пространственного объекта.

String[]

Массив Lua, содержащий ассоциированные идентификаторы информации.

Параметры:

spatialID: string

Используется хостом для уникальной идентификации пространственных объектов.

associationCode: string

Код запрашиваемой ассоциации, определенный каталогом фичеров.

roleCode: string или *nil*

Код запрашиваемой роли, определенный каталогом фичеров. Может быть ноль, если *associationCode* сам по себе достаточен для указания ассоциации или если все роли, определенные *associationCode*, желательны.

Примечания:

При вызове хост возвращает массив, содержащий идентификаторы информации для данной пространственной реализации, которая соответствует *associationCode* и *roleCode*. Если ассоциация информации не действительна для этого фичера в соответствии с каталогом фичеров, то хост возвращает ноль. Если совпадения не найдены, то хост возвращает пустой массив.

roleCode может быть ноль, в этом случае только *associationCode* следует использовать для поиска.

13-8.2.1.14 string[] HostSpatialGetAssociatedFeatureIDs(string *spatialID*)

Возвращаемое значение:

string[]

Массив Lua, содержащий запрошенные идентификаторы ассоциированных фичеров для пространственных данных, определяемых *spatialID*.

Nil

Нет фичеров, ассоциированных с *spatialID*.

Параметры:

spatialID: string

Используется хостом для уникальной идентификации пространственных объектов.

Примечания:

При вызове хост возвращает массив всех реализаций фичеров, которые ссылаются на заданный пространственный объект. Реализация фичера считается ассоциированной с пространственным объектом либо непосредственно через пространственные ассоциации с ним, либо косвенно в случае кривых, на которые ссылаются композитные кривые.

13-8.2.1.15 string[] HostInformationTypeGetSimpleAttribute(string informationTypeID, path path, string attributeCode)

Возвращаемое значение:

string[] или *nil*

Текстовое представление каждого значения атрибута, как описано в пункте 13-8.1. Массив возвращается, даже если атрибут имеет одно значение. Узел должен возвращать значение *nil*, если запрошенный атрибут отсутствует.

Параметры:

informationTypeID: string

Используется хостом для уникальной идентификации реализации информации.

path: path

Путь к атрибуту, как определено в пункте 13-6.2.

attributeCode: string

Один из кодов атрибутов, определенных в каталоге фичеров для типа информации, идентифицируемого в *informationTypeID*.

Примечания:

Поручает узлу выполнить простой поиск атрибута по атрибуту *attributeCode* по указанному пути для реализации информации, идентифицированной *informationTypeID*. *Nil* возвращается, если запрошенный атрибут отсутствует.

13-8.2.1.16 integer HostInformationTypeGetComplexAttributeCount(string informationTypeID, path path, string attributeCode)

Возвращаемое значение:

integer

Число соответствующих комплексных атрибутов, существующих на пути к реализации информации.

Параметры:

informationTypeID: string

Используется хостом для уникальной идентификации реализации информации.

path: path

Путь к атрибуту, описанный в пункте 13-6.2.

attributeCode: string

Один из кодов атрибутов, определенных в каталоге фичеров для типа информации, идентифицируемого *informationTypeID*.

Примечания:

Предписывает узлу возвращать число атрибутов, соответствующих *attributeCode*, по заданному пути к атрибуту для данной реализации информации. Данный путь всегда будет действительным для реализации информации. Возвращаемое целое число может быть нулём.

13-8.2.2 Функции доступа к типу информации

Эти функции позволяют среде скриптинга запрашивать тип информации для любого объекта из любого набора данных. Тип информации, предоставляемый хостом, должен соответствовать информации из соответствующего каталога фичеров.

13-8.2.2.1 string[] HostGetFeatureTypeCodes()

Возвращаемое значение:

string[]

Массив, содержащий все коды типов фичеров, определенные в каталоге фичеров.

Примечания:

13-8.2.2.2 string[] HostGetInformationTypeCodes()

Возвращаемое значение:

string[]

Массив, содержащий все коды типов информации, определенные в каталоге фичеров.

Примечания:

13-8.2.2.3 string[] HostGetSimpleAttributeTypeCodes()

Возвращаемое значение:

string[]

Массив, содержащий все коды типов простых атрибутов, определенные в каталоге фичеров.

Примечания:

13-8.2.2.4 string[] HostGetComplexAttributeTypeCodes()

Возвращаемое значение:

string[]

Массив, содержащий все коды типов сложных атрибутов, определенные в каталоге фичеров.

Примечания:

13-8.2.2.5 string[] HostGetRoleTypeCodes()

Возвращаемое значение:

string[]

Массив, содержащий все коды типов комплексных атрибутов, определенные в каталоге фичеров.

Примечания:

13-8.2.2.6 string[] HostGetInformationAssociationTypeCodes()

Возвращаемое значение:

string[]

Массив, содержащий все коды типов ассоциаций информации, определенные в каталоге фичеров.

Примечания:

13-8.2.2.7 string[] HostGetFeatureAssociationTypeCodes()

Возвращаемое значение:

string[]

Массив, содержащий все коды типов ассоциаций фичеров, определенные в каталоге фичеров.

Примечания:

13-8.2.2.8 FeatureType HostGetFeatureTypeInfo(string featureCode)

Возвращаемое значение:

FeatureType

Структура данных Lua, созданная функцией *CreateFeatureType()*.

Параметры:

featureCode: string

Код фичера, соответствующий записи в каталоге фичеров.

Примечания:

13-8.2.2.9 InformationType HostGetInformationTypeInfo(string informationCode)

Возвращаемое значение:

InformationType

Структура данных Lua, созданная функцией *CreateInformationType()*.

Параметры:

informationCode: string

Код информации, соответствующий записи в каталоге фичеров.

Примечания:

13-8.2.2.10 SimpleAttribute HostGetSimpleAttributeTypeInfo(string attributeCode)

Возвращаемое значение:

SimpleAttribute

Структура данных Lua, созданная функцией *CreateSimpleAttribute()*.

Параметры:

attributeCode: string

Код простого атрибута, соответствующий записи в каталоге фичеров.

Примечания:

13-8.2.2.11 ComplexAttribute HostGetComplexAttributeTypeInfo(string attributeCode)

Возвращаемое значение:

ComplexAttribute

Структура данных Lua, созданная функцией *CreateComplexAttribute()*.

Параметры:

attributeCode: string

Код комплексного атрибута, соответствующий записи в каталоге фичеров.

Примечания:

13-8.2.2.12 Role HostGetRoleTypeInfo(string roleCode)

Возвращаемое значение:

Role

Структура данных Lua, создаваемая функцией *CreateRole*.

Параметры:

roleCode: string

Код роли, соответствующий записи в каталоге фичеров.

Примечания:

13-8.2.2.13 InformationAssociation HostGetInformationAssociationTypeInfo(string informationAssociationCode)

Возвращаемое значение:

InformationAssociation

Структура данных Lua, создаваемая функцией *CreateInformationAssociation*.

Параметры:

informationAssociationCode: string

Код ассоциации информации, соответствующий записи в каталоге фичеров.

Примечания:

13-8.2.2.14 FeatureAssociation HostGetFeatureAssociationTypeInfo(string featureAssociationCode)

Возвращаемое значение:

FeatureAssociation

Структура данных Lua, создаваемая функцией *CreateFeatureAssociation*.

Параметры:

featureAssociationCode: string

Код ассоциации фичеров, соответствующий записи в каталоге фичеров.

Примечания:

13-8.2.3 Функции пространственных операций

Эти функции позволяют среде скриптинга выполнять реляционные тесты и операции на пространственных элементах.

Хост должен реализовать функции, описанные на следующих страницах, чтобы обеспечить среду скриптинга возможностью связывать пространственные объекты друг с другом.

13-8.2.3.1 boolean HostSpatialRelate(string *spatialID1*, string *spatialID2*, string *intersectionPatternMatrix*)

Возвращаемое значение:

boolean

Возвращает значение *True*, если геометрия, представленная двумя пространственными объектами, связана, как указано в матрице DE-9IM.

Параметры:

spatialID1: string

Используется хостом для уникальной идентификации пространственной реализации.

spatialID2: string

Используется хостом для уникальной идентификации пространственного экземпляра.

intersectionPatternMatrix: string

DE-9IM матрица пересечений, выраженная как девять символов в порядке большой строки. Например, при проверке перекрытия между двумя областями: "T*T***T**"

Примечания:

Пространственные объекты соотносятся с геометрией, представленной в *spatialID1* и *spatialID2* с использованием пересечения DE-9IM, указанного через строку *intersectionPatternMatrix*.

Подробнее о строковом представлении DE-9IM читайте в ISO 19125-1:2004, *Geographic information -- Simple feature access -- Part 1: Common architecture, section 6.1.14.2 The Dimensionally Extended Nine-Intersection Model (DE-9IM)*.

13-8.2.4 Функции поддержки отладки

Эти функции позволяют среде скриптинга взаимодействовать с отладчиком, который может работать на хосте. Отладчик может быть незаменимым помощником в разработке требуемых стандартных функций хоста.

Реализация функций поддержки отладчика хостом является необязательной. Скриптинг будет выполняться нормально независимо от того, реализует ли хост эти функции.

13-8.2.4.1 void HostDebuggerEntry(string *debugAction*, variant *parameters*)

Возвращаемое значение:

Нет

Параметры:

debugAction: string

Указывает требуемое действие отладчика:

break – Приостановить выполнение скрипта.

trace – Отобразить строку в отладочной консоли.

start_performance – Начать построчное профилирование кода скрипта.

stop_performance – Остановить построчное профилирование кода скрипта. Имя счетчика производительности находится в первом параметре.

first_chance_error – Уведомляет хост о предстоящем вызове функции ошибки в сценарии. Параметр – это сообщение, переданное функции ошибки. Второй параметр - глубина, передаваемая функции ошибки.

parameters: variant

Ноль или более параметров для использования при отладке.

Примечания:

Выполнение этой функции хостом является факультативным.
