АПК УИСС «ПАЛЛАДА»

СС-скрипты

Руководство администратора.

Компания «НЕВО-АСС»
Санкт-Петербург
2013

Дата публикации и версия программы
Опубликовано 10 Июня 2013 года.

Оглавление

Введение

Call Control скрипты предназначены для гибкого управления обработкой вызовов. Представляют собой программы, с помощью которой администратор может ветвить алгоритм по своему усмотрению.

Организация алгоритма обработки вызовов осуществляется на закладке «АЛГОРИТМЫ»[1] приложения «Конфигурация системы»

Call Control скрипт вставляется в точку типа "**CC -Скрипт"**[2]. Алгоритм, попав в точку такого типа, выполняет скрипт, и получив код возврата, выполняет ветвление, соответствующее этому коду. Скрипт также позволяет изменять системные и пользовательские переменные и ветвиться на основании анализа этих переменных.
Конструкции языка и переменные имеют чувствительность к регистру.

Ошибки в CC-скрипте приводят к прерыванию обработки текущего вызова.

См. также:
Business interface service *(далее BIS)*[3] предназначен для стороннего контроля за обработкой вызовов, а также для стороннего управления обработкой вызовов и инициализации исходящей связи.
При построении своей системы, заказчик может столкнуться с необходимостью придать ей функциональность, не предусмотренную разработчиками, или усовершенствовать её встроенные функции по своему усмотрению. Также может возникнуть потребность вывода информации об обработке вызовов на нестандартное оборудование или непредусмотренных методов её анализа и хранения.

В базовую поставку системы BIS не входит.

[1]Руководство администратора. Настройка системы. Раздел "Алгоритмы".

[2]Руководство администратора. Алгоритмы обработки вызовов. Раздел "CC -Скрипт".

[3]Руководство администратора. Business interface service.

1 Типы данных

Между типами допустимы неявные преобразования.
Предусмотрены следующие типы данных:

int

- целочисленная 4-байтовая переменная со знаком

float

- 8-байтовая переменная с плавающей точкой
(соответствует базовому типу double)

string

- строковая переменная переменной длины, заканчивающаяся NTS.
ВНИМАНИЕ: Длинна строки не может превышать 255 символов.

datetime

- Дата/время в формате unix
(количество секунд, прошедших с 01.01.1970 00:00:00)

2 Константы

Константы задаются в следующем формате:

int - целое число. Примеры: 123 - десятичное число; 0x123 - шеснадцатиричное число; 0123 - восьмеричное число.

float - число с плавающей точкой. Пример: 123.45

string - задаётся как строка, заключённая в двойные кавычки. Пример: "Hello world"

datetime - такую константу явно задать нельзя.
Можно использовать неявные преобразования от целочисленного значения или от строки в формате "dd.MM.yyyy hh:mm:ss"

3 Преобразования типов данных

Между типами при отработке операторов могут происходить неявные преобразования.
*Например: При выполнении операции присвоения, если источник имеет формат **float**, а приемник - формат **int**, то результатом будет целая часть от **float**.*

К типу	От типа	Преобразование
int	float	Отсекание дробной части
	string	Преобразование из строки. *Пример 1: I="100" - результат будет 100* *Пример 2: I="1abc" - результат будет 1* *Пример 3: I="abc" - результат будет 0*
	datetime	значение не изменяется
float	int	просто смена типа
	string	Преобразование из строки. *Пример 1: I="21.3" - результат будет 21.3* *Пример 2: I="21" - результат будет 21.0* *Пример 3: I="21.3abc" - результат будет 21.3* *Пример 4: I="abc" - результат будет 0.0*
	datetime	значение не изменяется
string	int	Преобразование к строке. *Пример: S=10 результат - "10"*
	float	Преобразование к строке. Четыре знака после точки. *Пример: S=10.1 результат - "10.1000"*
	datetime	Преобразование к строке в формате dd.MM.yyyy hh:mm:ss.
datetime	int	значение не изменяется
	float	Отсекание дробной части
	string	Возможно только от строки формата dd.MM.yyyy hh:mm:ss. Если формат строки иной, то исполнение скрипта будет прервано.

Дополнительная информация о допустимости преобразований типов описана в разделе операторы в колонке "типы".

4 Переменные

Скрипты могут использовать переменные, заранее определённые администратором.

- Переменная характеризуется **типом** и **именем**.
- **Длина имени** не может превышать 32 символов включая NTS.
- **Имя переменной** может содержать символы английского алфавита в верхнем и нижнем регистре, а также цифры и подстрочный символ "_".
- Имя не может начинаться с цифры.
- Имя переменной чувствительно к регистру.

Переменные делятся на следующие типы:

- **Системные переменные**.
- **Динамические переменные**
- **Статические переменные**

Любая переменная может быть воспроизведена в точке типа "**Воспроизведение**"[4].

4.1 Системные переменные

- предопределены разработчиками. Их имя начинается с символа @.
На данный момент определены следующие переменные:
(см. также "Типы данных")

4.1.1 @ACCUM

Строковая
- аккумулятор. Предназначен для сохранения строковой информации, накапливаемой в ходе вызова. Например: для сохранения какого-либо номера, набранного абонентом в точке «Накопление номера».

4.1.2 @CedAddr

Строковая
- Вызываемый номер.
При входящем вызове – номер, который встречная сторона использовала для вызова системы.
При исходящем вызове – номер, который система использует для вызова встречной стороны

4.1.3 @CingAddr

Строковая
- Вызывающий номер.
При входящем вызове – номер, вызывающего абонента.
При исходящем вызове – номер, который система передаёт встречной стороне как вызывающий.

4.1.4 @GMTIME

Дата/время
- время системы по 0-му меридиану

4.1.5 @LASTRECORDEDFILE

Строковая
Имя последнего, записанного файла аудиофайла.

4.1.6 @LOCALTIME

Дата/время
- локальное время системы

4.1.7 @VSPCLUSTERNO

Строковая

[4]Руководство администратора. Алгоритмы обработки вызовов. Раздел "Воспроизведение".

Используется в предустановленном алгоритме подсистемами конференции и оповещения для идентифицирующего номера кластера.

4.1.8 @VSPCONF

Строковая
Используется в предустановленном алгоритме подсистемой конференции для идентифицирующего номера конференции.

4.1.9 @VSPConfAccessCode

Строковая
Используется в предустановленном алгоритме подсистемой конференции для кода доступа к конференции.

4.1.10 @VSPDefTA

Строковая
Используется в предустановленном алгоритме подсистемами конференции и оповещения для задания транспортного адреса (H.323/SIP) встречной стороны по умолчанию.

4.1.11 @VSPLocalTA

Строковая
Используется в предустановленном алгоритме подсистемами конференции и оповещения для задания транспортного адреса (H.323/SIP) локальной стороны по умолчанию. Если не задан, используется IP-адрес первого подключённого адаптера.

4.1.12 @VSPNOTIFY

Строковая
Используется в предустановленном алгоритме подсистемой оповещения для кода доступа к конференции.

4.1.13 @VSPPIN

Строковая
Используется в предустановленном алгоритме подсистемами конференции и оповещения для PIN

4.1.14 @VSPReplacingNotifyFile

Строковая
Используется в предустановленном алгоритме подсистемой оповещения для задания имени файла для перезаписи сообщения оповещения.

4.1.15 @VSPSecToStart

Целочисленная
Используется в предустановленном алгоритме подсистемой конференции для хранения секунд до старта плановой конференции.

4.2 Динамические переменные

переменные, определённые администратором.
Их время жизни - один сеанс. В начале сеанса такие переменные инициализируются указанным значением. По окончании сеанса - уничтожаются. Для разных сеансов набор динамических переменных независим.

4.3 Статические переменные

переменные, определённые администратором.

Их время жизни не ограничено. При старте системы - инициализируются указанным значением. При присвоении, это значение переписывается. Все сеансы в системе пользуются одним набором статических переменных.
Пример:
переменная для переключения режима день/ночь

5 Операторы

Операторы позволяют выполнять действия с переменными и константами. Далее приведён список допустимых операторов.

Оператор	Приоритет	Описание	Типы
( )	-	Скобки. Явно указывают приоритет операций в выражении	
=	1	Присвоение. Операнд справа принимает значение от операнда справа.	Оба операнда: **int**,**float**,**string**, **datetime** Результат по типу первого операнда.
||	2	Логическое или	Допустимы все типы. См. примечания под таблицей.результат - **int**
&&	2	Логическое и	
|	3	Бинарное или	Оба операнда **int** или **datatime**. **datatime** приводится к **int** результат - **int**
^	3	Бинарное исключающее или	
&	3	Бинарное и	
<=	4	Меньше или равно	Оба операнда **string** или оба операнда **int**, **float**, **datatime**.результат - **int** См. примечания под таблицей.
>=	4	Больше или равно	
==	4	Равно	
!=	4	Не равно	
<	4	Меньше	
>	4	Больше	
+	5	Плюс	Оба операнда **string** или оба операнда **int** , **float**, **datatime**.Если хотя бы один операнд - float, то результат - float. Иначе, по типу первого операнда.
-	5	Минус	Оба операнда **int**, **float**, **datatime**.Если хотя бы один операнд - float, то результат - float. Иначе, по типу первого операнда.
$	6	Получить строку <операнд1> без <операнд2> начальных символов Пример: "123456789" $ 5 == "6789"	<операнд1> - **string** <операнд2> - **int** результат - **string**
$$	6	Получить <операнд2> символов от начала строки <операнд1> Пример: "123456789" $$ 5 == "12345" "123456789" $ 5 $$ 2 == "67"	
#	6	Получить строку <операнд1> без <операнд2> последних символов Пример: "123456789" # 5 == "1234"	
##	6	Получить <операнд2> символов от конца строки <операнд1> Пример: "123456789" ## 5 == "56789"; "123456789" # 5 ## 2 == "34"	
\	7	Найти в строке <операнд1> строку, начинающуюся с со строки <операнд2>	<операнд1> - **string** <операнд2> - **string** результат - **string**
*	7	Умножить	Оба операнда **int** , **float**, **datatime**.Если хотя бы один
/	7	Делить	

Оператор	Приоритет	Описание	Типы
%	7	Остаток от деления	операнд - float, то результат - float. Иначе, по типу первого операнда.
-	8	Унарный минус	Один операнд. тип - **int**, **float** **datetime**. результат соответствует типу операнда
~	9	Инверсия *(унарный)*	Один операнд. тип - **int**, **datetime**. результат соответствует типу операнда
!	10	Логическое не *(унарный)*	Один операнд. тип - **int**, **datetime**. См. примечания под таблицей. результат соответствует типу операнда

При выполнении логических операций как TRUE рассматривается строка, содержащая хотя бы один символ, или ненулевое значение для остальных типов. Как FALSE рассматривается пустая строка или нулевое значение для других типов.
Результат логических операций - 0*(FALSE)* или 1*(TRUE)*

6 Выражения

Выражения состоят из операндов и операторов. Для явного указания приоритетов используются круглые скобки.
Пример:
2+3*4 - результат 14
(2+3)*4 - результат 20
Выражение может использоваться внутри оператора if или в присвоении.

7 Конструкции языка

На данный момент реализованы следующие конструкции:

- **блок**
- **присвоение**
- **if-else**
- **return**
- **комментарий**
- **trace**

Конструкции языка отделяются друг от друга символом ";". После блока символ ";" ставить необязательно.
Новая строка не должна разделять на две части ключевые слова *(if,else,return)*, а также имена переменных и константы.

7.1 блок

Блок используется для выделения фрагмента исходного текста скрипта с целью указать, что этот фрагмент необходимо выполнить весь.
Пока применяется только в конструкции if-else.

7.2 присвоение

Конструкция присвоения предназначена для присвоения переменной результата выполнения выражения.
Синтаксис:
<переменная>= выражение>
Следует отметить, что сама по себе конструкция присвоения является выражением, в начале которого стоит переменная, а на втором месте - оператор "=".

7.3 if-else

Ветвление по условию.
Синтаксис:
if(< выражение>)<конструкция1>
[else<конструкция2>]
В случае, если строковый результат выражения не пуст, а для других типов не равен нулю, то выполняется <конструкция1>.
В противном случае, при наличии необязательной части "else"<конструкция2>, выполняется <конструкция2>.

7.4 return

Конструкция return выполняет возврат из скрипта с указанным кодом возврата. По этому коду возврата точка типа CC-скрипт выполнит по соответствующий ему переход.
Если при выполнении скрипта не встретилось ни одной конструкции return, то код возврата полагается равным нулю.
Синтаксис:
return <выражение >
Выражение должно давать целочисленный результат. Если результат выражения имеет другой тип, то происходит его неявное преобразование к int.

7.5 комментарий

Комментарий игнорируется при выполнении скрипта. Служит для словесного описания действий, выполняемых скриптом. Может находиться в любом месте программы. Комментарий не должен разрывать конструкции языка и имена переменных.
Синтаксис:
/* <текст>*/

7.6 trace

Конструкция trace предназначена для отладочного вывода переменных и выражений в журнал Commutator.log.
(Log –файлы хранятся в каталоге <Каталог установки>\palladad\Logs.)
Синтаксис:
trace < выражение>
Отладочный вывод попадает в журнал output.log

8 ПРИЛОЖЕНИЕ

Пример:

Администратором объявлена целочисленная переменная AInt со значением 0; Существует следующий скрипт:

```
/*Ограничение количества воспроизведения информации */
if(AInt<3){
        /*Менее трёх вхождений в скрипт, позволяем прослушать информацию */
        AInt=AInt+1;
        return 1;
}else{
        /*Более трёх вхождений в скрипт, абонент похоже балуется, сообщим ему, */
        /*что так делать не стоит */
        return 2;
}
```

Вставив такой скрипт перед воспроизведением какой либо информации, для которой по алгоритму возможен повтор, Вы защитите систему от непроизводительного занятия ресурсов. По коду возврата 1 позволяем прослушать информацию, по коду возврата 2 - воспроизведём абоненту сообщение "Извините, количество попыток ограничено" и отобьём его.