DGScript

Introduction

In addition to visual programming blocks, DGLux also provides the ability to write custom script that interacts with the client-side environment. DGSript is utilized in several areas of the application.

The “DGScript” Non-Visual Component provides the ability to define custom “Dynamic Properties” that get passed in as variables that are accessible by name from the script property.

Examples:

```
//Simple Equality

if (property1 >= property2)
{
    output = “Greater”;
}
else
{
    output = “Less”;
}
```

```
//Concatenate all “Dynamic Properties” of a DGScript block

var dProperties = parent.parent.dynamicProperties;
var tStr = "";

for (var i=1; i < dProperties.length; i++)
    //output property is first so we want to skip it
{
    var name = dProperties[i].name;
    var value = dProperties[i].value;

    tStr += value;
}

output = tStr;
```

```
//Table Property Operations

getTableRows(name:String, sheetId:int, makecopy:Boolean = true):Array
getTableColumns(name:String, sheetId:int):Array
setTableColumns(name:String,sheetId:int, arrayofStrings:Array):Boolean
```

```
setTableRows(name:String, sheetId:int, arrayOfObjects:Array):Boolean
```

The “Table” data type supports a formula column that contains a script. All columns in the current sheet are accessible by name in the script.

Examples:

```
(columnA + columnB) / 2;
```

```
if (columnA >= columnB)
{
    return true;
}
else
{
    return false;
}
```

```
if (row == 0) //Execute only once
{
    var runningTotal = 0;
    for (var i=0; i < rows.length; i++) //rows is an Array of all rows in the
sheet
    {
        runningTotal += rows[i].columnA;
    }
    return (runningTotal / rows.length);
}
else
{
    return null;
}
```

```
formatDate(string2Date(ts), "D-MMM-YYYY L:NN A")
```

The Language

The language supported by DGScript is essentially full AS3 with some minor differences. Its major features are listed here. This chapter explains the language features in detail. A full AS3 reference is available from Adobe at:
http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/package-detail.html

Supported Functionality

DGScript supports a powerful language, which is essentially a full actionscript3Script with some minor differences. The following list shows its major features:

1. Supports all the operators and flow control statements of actionscript3Script/AS3.
2. Supports all E4X operators and operations.
3. Supports user function declaration.
4. Supports the syntax of AS3-style type declarations for var and function return values and parameters. Type, however, is simply discarded.
5. Supports import statement to import AS3 classes so that they can be used to new instances or call its static methods.
6. All Flash top-level functions are available to dynamic code, and a number of built-in functions, including printf(),importFunction() and importStaticMethods().
7. Provides comprehensive interaction with the hosting AS3 environment, with support at both the language and the API levels.
8. Does not support actionscript3Script object-orientation afterthoughts such as prototype and actionscript3Script-style object constructors. In other words, the language does not support defining classes in any way. Object usage, i.e., the dot-notation, is supported.
9. Extended keyword support of and, or, not, xor, nand, and nor, useful for taking end-user input with more English-like logical expressions.

Variables

Variables do not have to be declared with var before use. All variables are dynamically typed; you can specify a type for a variable in var but that type is ignored.

Expressions

All AS3 expressions are supported, including literals and E4X expressions. Literal syntax includes null, this, boolean, integer, number, string, object, array and E4X objects. This references the current object scope on which DGScript is running. The super keyword in AS3 is not supported.

Extended Logical Operator Keywords

In addition to actionscript3Script/AS3 operators, the dynamic language supports these extended logical operator keywords: and, or,not, nand, nor, and xor. They are useful for taking end-user input with more English-like logical expressions:

```
var sick = false;
var young = true;
```

```
if (young and not sick) premium = "low";
```

Using ActionScript 3 Classes

Like in AS3, a class must first be imported via the import statement before it can be used. Unlike AS3, only fully qualified class names are accepted; no wildcard characters are allowed. Also, the AS3 classes to be used by the dynamic code must be present at runtime. You can import multiple classes in a single statement. Once a class is imported, it can be used just like in AS3. The class name without the package part is used; classes can be assigned to variables; you can create instances of a class, and call static methods of that class. For instance:

```
import flash.display.Sprite, mx.utils.StringUtil;

var mySprite:Sprite = new Sprite;

printf( StringUtil.trim('   abc   ') );
```

Statements

All AS3 statements are supported in the dynamic language with a handful of exceptions. The following table lists supported statements and directives.

ActionScript 3	DGScript	Comment
break	break	Same as AS3
case	case	Same as AS3
continue	continue	Same as AS3
default	default	Same as AS3
do..while	do..while	Same as AS3
else	else	Same as AS3
for	for	Same as AS3
for..in	for..in	Same as AS3
for each..in	for each..in	Same as AS3 except that, when var is used to declare the variable, type is not allowed; this is different from the rest of the language.
if	if	Same as AS3
return	return	Same as AS3
switch	switch	Same as AS3 and actionscript3Script; the case values can be any expressions and are not necessarily constants.
throw	throw	Same as AS3
try..catch..finally		Not supported.
while	while	Same as AS3

ActionScript 3	DGScript	Comment
with		Not supported.
default xml namespace	default xml namespace	Same as AS3
import	import	Supported to import AS3 classes that are avaiable at the time of running.
include		Not supported.
use namespace		Not supported.

User-Defined Functions

Functions can be defined dynamically, too. The syntax is exactly the same as in AS3, but the type information for parameters and return type is simply discarded. But notice! These functions are not the same as AS3 functions! Therefore, they can not be returned to the calling AS3 code nor be passed as parameters to imported AS3 functions (see below). User-defined functions can be assigned to variables or passed as parameters to other user-defined functions, like so:

```
function inc(x) {
  return x+1
}
function dec(x) {
  return x-1
}
function delegate(x:int, f:Function):int {
  return f(x)
}

delegate(5, dec)
```

Using AS3 Top-Level Classes and Functions

Flash top-level classes and functions can be used exactly like in AS3. For example:

```
var date = new Date(1987, 3, 5);

var value = escape('cond=age<50 and age>30');

var num = Number('1234');

trace('TRACE from dynamic code.');
```

Built-in Functions

The dynamic language includes a few built-in functions.

```
function printf(msg_fmt, ...):void
```

This function prints the message to the system's output display. The first parameter can be a string containing parameter indicators like {0}, {1}, etc.; they are replaced by values of the following parameters.

```
function importFunction(name, theFunction):void
```

This function imports a function, typically a class static method, to be globally invokable. For instance:

```
import mx.utils.StringUtil;
importFunction('trim', StringUtil.trim);

printf( trim('    abc    ') ); // call the imported function
```

```
function importStaticMethods(cls, criteria:*=null):void
```

This function imports all the static methods of cls, a class object, that match the criteria. If criteria is null, all static methods are returned; otherwise, the criteria can be either a RegExp object or an array of strings. For instance:

```
importStaticMethod(Math, [ 'sin', 'cos' ]);

printf( sin(0.12) ); // call the imported function
```