
Slingshot JavaScript Interface

Slingshot Version 2

Document Version 1.1
Dec 2007

For more information contact:

Swissrisk Financial Systems GmbH
Technical Support
Holzhausenstrasse 44
D-60322 Frankfurt
Germany

Phone +49 69 50952 111
Fax: +49 69 50952 199
Email hotline@sr-financial-systems.com
Website www.sr-financial-systems.com

Copyright Information

This document is protected by copyright law and may not be reproduced or distributed either in part or in total. The licensee is not allowed to pass on the software or the accompanying written materials to third parties or make them otherwise available without prior written agreement of the licensor. Information in this document that refers to possible product extensions or to available accessories is not legally binding, especially because the products are subject to continuous adaptation and because the information may also relate to future development. The contents of this document can change without prior notice and does not represent any legal obligation on the part of Swissrisk Financial Systems GmbH.

Swissrisk Financial Systems GmbH cannot be made liable for the correctness of information in this document nor for damages resulting from the use of this information or the impossibility of using this information. All other legal regulations for using the software and the corresponding documentation are set in the applicable license agreement.

Slingshot is a trademark of Swissrisk Financial Systems GmbH. All other product and company names mentioned in this manual are trademarks of their respective companies

Published by:

Swissrisk Financial Systems GmbH
Holzhausenstrasse 44
D-60322 Frankfurt
Germany

Phone: +49 69 50952-0
Fax: +49 69 50952-333
Website: www.sr-financial-systems.com

Copyright © Swissrisk Financial Systems GmbH | All rights reserved

Slingshot JavaScript Interface

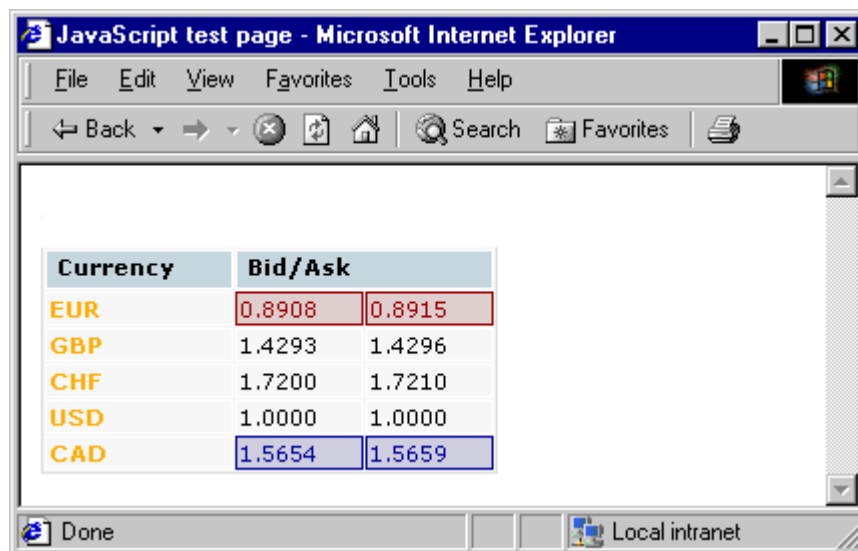
Introduction	1
Implementation Details	1
Starting the Java Applet	2
APPLET Tags.....	2
PARAM Tags.....	2
The Interface Methods	3
Sample JavaScript Program	4
Introduction.....	4
sample_quotelist.html.....	5
getData.js.....	8
highlights.js.....	10
ss_jsap_style.css.....	12
JavaScript Callbacks	13
PARAM Tags for Callbacks.....	13
Example.....	14

Introduction

This document outlines the Slingshot JavaScript Interface. This interface is used to provide the latest real-time updates to JavaScript Programmers, allowing them to display the data anywhere on a web page.

It is very important to understand the difference between Java and JavaScript before continuing. In this instance it is the JavaScript which calls the Java Applet. Swissrisk Financial Systems GmbH provides the Java Applet but the JavaScript code must be written by the end user. Sample JavaScript code is provided at the end of this document.

It is of course required that the users browser is both Java and JavaScript enabled.

A screenshot of a Microsoft Internet Explorer browser window. The title bar reads "JavaScript test page - Microsoft Internet Explorer". The address bar shows "Local intranet". The main content area displays a table with two columns: "Currency" and "Bid/Ask". The table contains five rows of data for different currencies: EUR, GBP, CHF, USD, and CAD. The EUR row has bid and ask rates of 0.8908 and 0.8915 respectively. The CAD row has bid and ask rates of 1.5654 and 1.5659 respectively. The table is styled with a light blue header and alternating row colors.

Currency	Bid/Ask	
EUR	0.8908	0.8915
GBP	1.4293	1.4296
CHF	1.7200	1.7210
USD	1.0000	1.0000
CAD	1.5654	1.5659

Figure 1.1

Implementation Details

The purpose of this Java Applet is to allow any JavaScript enabled browser access to real time data.

To achieve this a Java Applet is provided, in the file "JScript.zip", which uses the Slingshot ClientAPI to request real-time data (or you can use "SLSALL.zip" which contains all the Java applets). This Java Applet provides the ability for JavaScript to call methods to receive the latest data for display on screen, or any other use.

There are two possible ways to use the JavaScript Applet. The first is polling, and the second is JavaScript Callbacks which are detailed at the end of this document.

In the first instance, the Java Applet is polled by JavaScript code. If new data is available the JavaScript can then take that data from the Java Applet. The frequency of the polling is up to the JavaScript writer but it is not recommended that this frequency be less than 10 milliseconds. The polling can be as infrequent as you wish because the internal memory management ensures that only the latest updates are stored in memory and then returned to the JavaScript.

The Java Applet maintains the details of the latest updates to all the objects in its internal cache. The Java Applet only provides the JavaScript with the **latest** available data. Incoming data is automatically merged on the client side (in the Java Applet) so that for every successful poll only the latest data is provided to the JavaScript.

This process means that updates to different objects may not be received in the order in which they are sent. It also means that all messages passed to the JavaScript contain the latest updates for all the FID's which were requested.

Starting the Java Applet

The Java Applet is started in the HTML page via the standard HTML applet tag. A sample Applet tag is available below.

```
<APPLET WIDTH=1 HEIGHT=1 CODE="net.Slingshot.JScript.JScript"
Archive="JScript.zip" CodeBase="http://127.0.0.1" Name="JSApplet">
<PARAM Name="Server" Value="http://127.0.0.1:80">
<PARAM Name="SErViCe" Value="MyDataFeed">
<PARAM Name="Objects" Value="EUR=,GBP=,DEM=,IEP=,AUD=">
<PARAM Name="fids" Value="3,5,22,25,115">
<PARAM Name="ApiDebug" Value="TRUE">
<PARAM Name="Debug" Value="TRUE">
</APPLET>
```

APPLET Tags

- **WIDTH:** This should be 1
- **HEIGHT:** This should be 1
- **CODE:** This should be "net.Slingshot.JScript.JScript"
- **Archive:** This should be "Jscript.zip", unless a signed applet is used
- **CodeBase:** This should be the address and port of the machine that is serving the Applet. This is usually the WDS.
- **Name:** This should be a unique name within this web page which the JavaScript Code will refer to this Object as.

PARAM Tags

- **Server:** The URL of the Server. This should contain the port number too.
- **Service:** The Name of the Service on the Server. Only one Service can be defined per applet.
- **Objects:** A comma-delimited list of the objects to be requested.
- **FIDs:** A comma-delimited list of Field ID's to request for each and every object.
- **Debug:** Whether or not to display debug information of the applet in the Java Console.
- **ApiDebug:** Whether or not to show debug information in the java Console for Slingshot client API.

- **AuthorisationID:** The ID to be supplied by the API to WDS if the authorisation method is AUTHORISATION_ID.
- **CacheEnabled:** Set when applet is run in polling mode. Default is off.
- **UseCookie:** Check if the user wants to use the document's cookie to get the AuthorisationID.

Note: See the JavaScript Callbacks section for more possible PARAM tags.

The Interface Methods

The interface provided, which the JavaScript can use, is defined below. It consists of constants and functions:

Constant Name	Value	Description
NO_DATA_AVAILABLE	0	No new data currently available.
LIVE_DATA_AVAILABLE	1	Live Data is available
DATA_STALE	2	The data is stale.

Some JavaScript programs may not want the complexity of implementing stale handling. In this case simply checking that pollForData() returns a value greater than 0 will suffice as a check for available data. Otherwise it is best to check the return value from pollForData() using the defined constants.

Returns	Name	Description
int	pollForData	This is the most important function call. It is used to check for new data. If this returns a value > 0 then new data is available by calling getObjectName() and getObjectData().
String	getObjectName	Returns the name of the latest Object to be updated.
String	GetObjectData (int FID)	Returns the latest data for the specified FID. Returns null if there is a problem. The Field number must be passed as a parameter to this function.
Void	insert (String inObjName, int FID, String data)	This function allows you to contribute data to an Object.
Void	request (String recordName, String FidList)	Request a new field of a (new or existing) record
Void	release (String recordName)	Release all fields of one record
Void	releaseAll ()	Releases all objects (connection may be closed soon)

Sample JavaScript Program

Introduction

The sample JavaScript is quite complicated and includes features such as up/down highlight colours. Other features include the ability to call a JavaScript function when you click on the object name. Therefore a good knowledge of JavaScript is required to understand the code.

The example is made up of four files. The main file is "sample_quotelist.html". This contains the basis of the JavaScript code. One of its main purposes is to ensure that the Java Applet is fully loaded before the JavaScript is called. It also defines how the page will look. The file "getData.js" is used to get the data and display it on screen. The file "highlights.js" is used to control the highlighting functionality. The stylesheet "ss_jsap_style.css" defines how the page looks on screen.

sample_quotelist.html

```
<HTML>
<HEAD>
<TITLE>JavaScript test page</TITLE>
  <script language="JavaScript" src="getData.js"></script>
  <script language="JavaScript" src="highlights.js"></script>
  <script language="Javascript">
  ///////////////////////////////////////////////////////////////////
  /
  //
  // JavaScript to query Slingshot Java Applet.
  // The query can happen at configurable intervals.
  // When new data is found it is displayed using an html table.
  //
  // The application uses the functions getData(), highlight() and clearHighlight().
  // These are provided in the files getData.js and highlights.js
  ///////////////////////////////////////////////////////////////////
  /

  //list of RICs to be requested (delimited & terminating with a comma)
  var objects          = "EUR=,GBP=,CHF=,USD=,CAD="

  //default values
  //list of Field IDs (FIDs) to be requested (delimited with a comma)
  var fids             = "115,22,25";
  //Name of the service on the WDS which the Applet will connect to for its data
  var serviceName      = "MyDataFeed";
  //Name of the server on the WDS which the Applet will connect to for its data
  var hostName         = "http://127.0.0.1";

  //period for querying the slingshot applet (ms)
  var pollRate         = 50;
  //Highlight length in ms
  var highlightTimeout = 2000;

  //define colours
  var updateTextUpColourHex = "#000099"; //Text colour on highlight when up
  var updateTextDownColourHex = "#990000"; //Text colour on highlight when down
  var updateTextColourHex = "#000000"; //Default text colour on highlight
  var updateBgUpColourHex = "#ccccdd"; //Background colour on highlight when up
  var updateBgDownColourHex = "#ddcccc"; //Background colour on highlight when down
  var updateBgColourHex = "#66cc55"; //Default background colour on highlights

  var bgColourHex      = "#f4f4f4"; //Default background colour
```

```

var defaultTextColourHex    = "#000000"; //Default text colour

////////////////////////////////////
</script>

<link rel="STYLESHEET" type="text/css" href="ss_jsap_style.css">
</HEAD>
<BODY>

<script language="JavaScript">
    //call the applet.
    //use the variables defined above to call the fids and objects
    document.writeln('<APPLET WIDTH=1 HEIGHT=1 CODE="net.Slingshot.JScript.JScript"
Archive="JScript.zip" CodeBase="'+ hostName +':80" Name="JSApplet">');
    document.writeln(' <PARAM Name="Server" Value="'+ hostName +'>');
    document.writeln(' <PARAM Name="Service" Value="'+ serviceName +'>');
    document.writeln(' <PARAM Name="Objects" Value="'+ objects +'>');
    document.writeln(' <PARAM Name="fids" Value="'+ fids +'>');
    document.writeln(' <PARAM Name="ColourHex" Value="#EEEEEE">');
    document.writeln(' <PARAM Name="Debug" Value="FALSE">');
    document.writeln(' <PARAM Name="CacheEnabled" Value="TRUE">');
    document.writeln('</APPLET>');
</script>

<br>

<table id="currencyOut" bgcolor="#f4f4f4" bordercolor="#ffffff" border="1"
cellpadding="1" cellspacing="1" class="window">
<tr>
    <td valign="top" bgcolor="#C2D0D8"><font
class="tabletitle">&nbsp;Currency</font></td>
    <td valign="top" bgcolor="#C2D0D8" colspan="2"><font
class="tabletitle">&nbsp;Bid/Ask</font></td>
</tr>
<tr id="EUR_row"><td id="EUR" width="90"><a href="javascript:alert('Call a
function!')">EUR</a></td><td id="EUR22" width="60"></td><td id="EUR25"
width="60"></td></tr>
<tr id="GBP_row"><td id="GBP"><a href="javascript:alert('Call a
function!')">GBP</a></td><td id="GBP22"></td><td id="GBP25"></td></tr>
<tr id="CHF_row"><td id="CHF"><a href="javascript:alert('Call a
function!')">CHF</a></td><td id="CHF22"></td><td id="CHF25"></td></tr>
<tr id="USD_row"><td id="USD"><a href="javascript:alert('Call a
function!')">USD</a></td><td id="USD22"></td><td id="USD25"></td></tr>
<tr id="CAD_row"><td id="CAD"><a href="javascript:alert('Call a
function!')">CAD</a></td><td id="CAD22"></td><td id="CAD25"></td></tr>
</table>

<script language="JavaScript">

```

```
<!--  
  
    function startUpdates() {  
        //ensure the applet is ready to be polled  
        try {  
            getData(fids);  
        }  
        catch(e) {  
            //wait one second then try again  
            setTimeout("startUpdates()",1000);  
        }  
    }  
  
    //run the application  
    startUpdates();  
  
//-->  
</script>  
</BODY>  
</HTML>
```

getData.js

```

////////////////////////////////////
// getData(fids)
////////////////////////////////////
// Check for new data from the Slingshot Applet
//
// 'fids' is a list of FIDs to be requested (delimited with a comma)
// This function takes new data as it arrives and builds a reference to table cell
// from the name of the Object and the FID number which provided that data.
// The data is then displayed inn the corresponding table cell (which must be
// created including an 'id' attribute).
// For the purpose of displaying different colours on up/down ticks, the first
// FID should be a suitable change field (eg: change, or Tick).
// //////////////////////////////////////
////

function getData(fids) {
  //Get the data from the Slingshot Applet and write it to the screen in the
  //correct location.

  //If there is new data present...
  if(document.JSApplet.pollForData() > 0) {

    //split fids passed to function into an array.
    fidArray = fids.split(",");

    //get data from the applet and write into the array.
    //the updateArray will take thge form:
    // updateArray[0] = name of Object requested
    // updateArray[1] = data returned for first FID requested for this Object
    // updateArray[2] = data returned for second FID requested for this Object
    //
    // updateArray[n] = data returned for nth FID requested for this Object
    updateArray = new Array();

    updateArray[0] = document.JSApplet.getObjectName();

    for(f=1; f<(fidArray.length+1); f=f+1) {
      pos = parseInt(f); //convert the index variable to an integer
      m = pos-1; //transform the index value
      //put data from the applet into the updateArray.
      updateArray[f] = document.JSApplet.getObjectData(fidArray[m]);
    }

    //get the currency name for reference in the table

```

```
//This operation is set for currency data from a Bridge feed
//ricIDArray = updateArray[0].split("$$");
//ricID = ricIDArray[1];

//This operation is set for currency data from a Reuters feed
ricIDArray = updateArray[0].split("=");
ricID = ricIDArray[0];

//cycle through fields supplied and build the correct table cell id.
for(i in fidArray)  {

    tmp = parseInt(i);    //convert array index to integer
    n = (tmp+1);         //offset the array position in the update array

    //Assume that the first FID is a tick reference (eg: change, or Tick).
    //If we are looking at the first fid, then convert it to a floating
    //point number.
    //This will be passed through to the Highlight function to facilitate
    //different up/down tick colours.
    if (i == 0){
        //tick = parseFloat(updateArray[1]) //used for Bridge data
        //used when U/D tick is available
        if (updateArray[1]=="U" || updateArray[1]=="u" ||
updateArray[1]=="1") tick=1;
        else if (updateArray[1]=="D" || updateArray[1]=="d" ||
updateArray[1]=="2") tick=-1;
        else tick = 0;
    }

    else {

        //build cell id from the object name plus the fid number.
        cellID = (ricID + fidArray[i]);

        row = ricID + "_row";

        //write the cell of the table
        //highlight(data,which row,which cell,up/down);
        highlight(updateArray[n],row,cellID,tick);
    }
}
}
setTimeout("getData('"+fids+"')", pollRate);
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

highlights.js

```
////////////////////////////////////
// highlight(data,whatRow,whatCell,tick)
////////////////////////////////////
// Update the value shown and highlight the appropriate cells or row of the table
//
// 'data' provides the data to be output to the screen
// 'whatRow' is an identifier which corresponds to the id tag of a table row.
//     this should be used when wanting to highlight the entire row.
// 'whatCell' is an identifier which corresponds to the id tag of a table cell.
//     this should be used when wanting to highlight a single cell.
// 'tick' is used to establish the up/down highlight colour.
// It can be either an Integer or Floating point value.
//
// Wherever one of the above parameters is not require, avalue of "null" should be
// passed to the function in its place.
//
////////////////////////////////////

function highlight(data,whatRow,whatCell,tick) {

    //set correct highlight colours depending on up/down state of last tick
    if(tick>0) {
        textCol = updateTextUpColourHex;
        bgCol = updateBgUpColourHex;
    }
    else if(tick<0) {
        textCol = updateTextDownColourHex;
        bgCol = updateBgDownColourHex;
    }
    else {
        textCol = updateTextColourHex;
        bgCol = updateBgColourHex;
    }

    //update text
    document.getElementById(whatCell).innerHTML="<font
color='"+textCol+"'>"+data+"</font>";

    //highlight background colour
    document.getElementById(whatCell).bgColor = bgCol;
    document.getElementById(whatCell).borderColor = textCol;

    //reset the bg color in a second
```

```
        setTimeout("clearHighlight('"+data+"','"+whatRow+"','"+whatCell+"','"+tick+"')",
highlightTimeout);
    }

/////////////////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////
///
// clearHighlight(data,whatRow,whatCell,tick)
/////////////////////////////////////////////////////////////////
// Clear the highlight colours and modify the text colour if appropriate
//
// 'data' provides the data to be output to the screen
// 'whatRow' is an identifier which corresponds to the id tag of a table row.
//     this should be used when wanting to highlight the entire row.
// 'whatCell' is an identifier which corresponds to the id tag of a table cell.
//     this should be used when wanting to highlight a single cell.
// 'tick' is used to establish the up/down highlight or persisting text colour.
//     It can be either an Integer or Floating point value.
//
// Wherever one of the above parameters is not require, avalue of "null" should be
// passed to the function in its place.
//
/////////////////////////////////////////////////////////////////

function clearHighlight(data,whatRow,whatCell,tick)    {

    //update text
    document.getElementById(whatCell).innerHTML="<font
color='"+defaultTextColourHex+"'+>"+data+"</font>";
    //highlight background colour
    document.getElementById(whatCell).bgColor = bgColourHex;
    document.getElementById(whatCell).borderColor = "ffffff";

}

/////////////////////////////////////////////////////////////////
```

ss_jsap_style.css

```
body, td {
    font-family: Verdana, sans-serif;
    font-size: 11px;
    color: #000000;
}

.window {
    border : 1px solid #DFDFDF;
    border-top : none;
}

.tabletitle
{
    FONT-FAMILY: Verdana,Arial,Helvetica;
    FONT-SIZE: 11px;
    font-weight: bold;
    COLOR: #000000;
    HEIGHT:16;
}

a { font-family: Verdana, sans-serif;
    font-size: 11px;
    font-weight: 600;
    color:ffaa00;
    text-decoration:none;
}

a:hover {
    text-decoration:underline;
}
```

JavaScript Callbacks

A second method of retrieving data from the JScript Applet is to use JavaScript Callbacks. This removes the need to poll the Applet, asking whether or not an update has been received. Instead, some JavaScript functions are registered with the JScript Applet and these are called when an update is received.

All parameters returned by the JavaScript Callback functions are in the form of Strings. Each function supplied must contain the correct number of parameters or an exception will be thrown.

The APPLET tag on the HTML page must contain the word "MAYSCRIPT", otherwise JavaScript Callbacks will not work.

Note: JavaScript Callbacks do **not** work with all browsers. For example, Netscape 6 does not support JavaScript Callbacks.

PARAM Tags for Callbacks

- **OnLoad:** This should contain the name of the function to be called when the Applet has completed loading. It is passed no parameters.
- **OnData:** This function is called when new data arrives. It contains three parameters. The first is the Objects name, the second is the number of the FID that has been updated and the last is the new value of the data.
- **OnState:** When there is a state change this function is called with one parameter, the new State of the Object. This parameter is a number and should be compared to the table below.

State Name	Value
HTTP_CONNECTION_INITIALISING	0
HTTP_CONNECTION_ACTIVE	1
HTTP_CONNECTION_STALE	2
HTTP_CONNECTION_ERROR	3
HTTP_CONNECTION_CLOSED	4
HTTP_MAX_CLIENTS_REACHED	5
HTTP_CONNECTION_UNATHORISED	6

Example

```
<HTML><HEAD>
<TITLE>JavaScript Callbacks</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function justLoaded()
{
alert("The Applet Has Finished Loading");
}

function  gotData(rec, fid, val, offset, state, flags)
{
alert("Object <"+rec+"> , FID "+fid+" has updated with the value
<"+val+">");
}

</SCRIPT>
</HEAD><BODY BGCOLOR=WHITE TEXT=BLACK>

<APPLET WIDTH=1 HEIGHT=1 CODE="net.Slingshot.JScript.JScript"
archive="JScript.jar" CODEBASE="http://127.0.0.1:80" MAYSCRIPT>
<PARAM Name=Server Value="http://127.0.0.1:80">
<PARAM Name=Service Value=MyDataFeed>
<PARAM Name=Objects Value="EUR=">
<PARAM Name=FIDs Value="22">
<PARAM Name=DEBUG Value=FALSE>
<PARAM Name=OnLoad Value="justLoaded">
<PARAM Name=OnData Value="gotData">
</APPLET>

<HR>
The Applet is above the line. A Dialog should appear when the applet
has finished loading....
<BR> Dialogs also appear when data arrives for EUR=
<BR> The MAYSCRIPT Tag must be included in the APPLETTAG tag.
<BR>This example works in Netscape 4.76 and IE 5.5
<BR>It does not work in Netscape 6 and is untested in other browsers.
</BODY></HTML>
```