Desktop Studio- Events

Version: 18.1

intellicus

Dated: August 2015

Acknowledgements

Intellicus acknowledges using of third-party libraries to extend support to the functionalities that they provide.

For details, visit: http://www.intellicus.com/acknowledgements.htm

Contents

1 Events

Report Events

Intellicus Studio processes the report by initializing the data sources, and then triggering events for report header section. This is followed by the page header section, report groups section, details section, report footer section, and the page footer section.

Intellicus Studio continues the processing while caching the pages in its internal page cache. The stream mode of output helps in viewing the first few pages of a long running report immediately.

Important: To achieve this, the report must not contain any forward reference, such as a Grand Summary field in the report header.

Intellicus Studio provides facility to customize the events using the Script Editor. Coding is done in JavaScript syntax, and is event based. The following table lists the events that are passed at report level.

Important: You need to make sure that the code pertaining to a particular event is written within the appropriate event only.

Event	Description
OnReportStart	This event is fired before report objects such as fields and sections are constructed. (Before the report starts to execute itself)
OnReportEnd	This event fires after execution of the report.
OnPageStart	This event is fired before displaying every page. This event does not ensure that the previous page's display has been completed.
OnPageEnd	This event is fired after the display of every rendering.
OnHyperlink (Button, Link)	This event is fired when the end user clicks on the hyperlink on the report output. The mouse button and the URL are passed in as parameters.
OnDataInitialize	This event is fired after the report is loaded, or SQL statement is fired or SQL fields are created. In this event, new report fields can be added and existing fields removed.
OnFetchData (eof)	This event is fired after each row of the report SQL statement is fetched from the database. In this event, the report field's data can be accessed for calculation and manipulation.
OnNoData	This event is fired when zero rows are fetched from the report SQL statement.
OnPrintProgress (PageNumber)	This event is fired when the printing progresses to next page. The printing process refers to sending the page data to the printer driver and not printing the page on the paper.
OnError (Number, Description, Scode, Source, HelpFile, HelpContext, CancelDisplay)	This event is fired when any error occurs while running the report. In this event, the error messages can be changed or can be suppressed.

Section Events

Intellicus passes three events at section level for all sections. The sequence of events depends on the summary objects and their section dependencies. The event sequence is:

Format

This event is fired after the data is loaded and bound to the fields, but before the section is laid out for printing. You can use this event to modify the layout of the section or any of the controls on it.

Note: This is the only event in which you can modify the height of the section.

BeforePrint

It is fired before the section is rendered to the 'Canvas' object. You can use this event to modify the values of the controls before they are printed. Any changes that are made here will not affect the height of the section.

Important: It is recommended, NOT to access any report fields in this event. If you need the value of a field in this event, you should use a hidden control to store the value temporarily in the format event.

AfterPrint

It is fired after the section is rendered to the 'Canvas' object. You can use this event to update any counters that you need to use after the report is completed.

2 Script Editor

Intellicus provides facility to write scripts for field properties and field events using the Script Editor. This enables you to define your own constructs for report generation. To get the **Script Editor** dialog box, click the option **Scripting** from **Tools** menu.

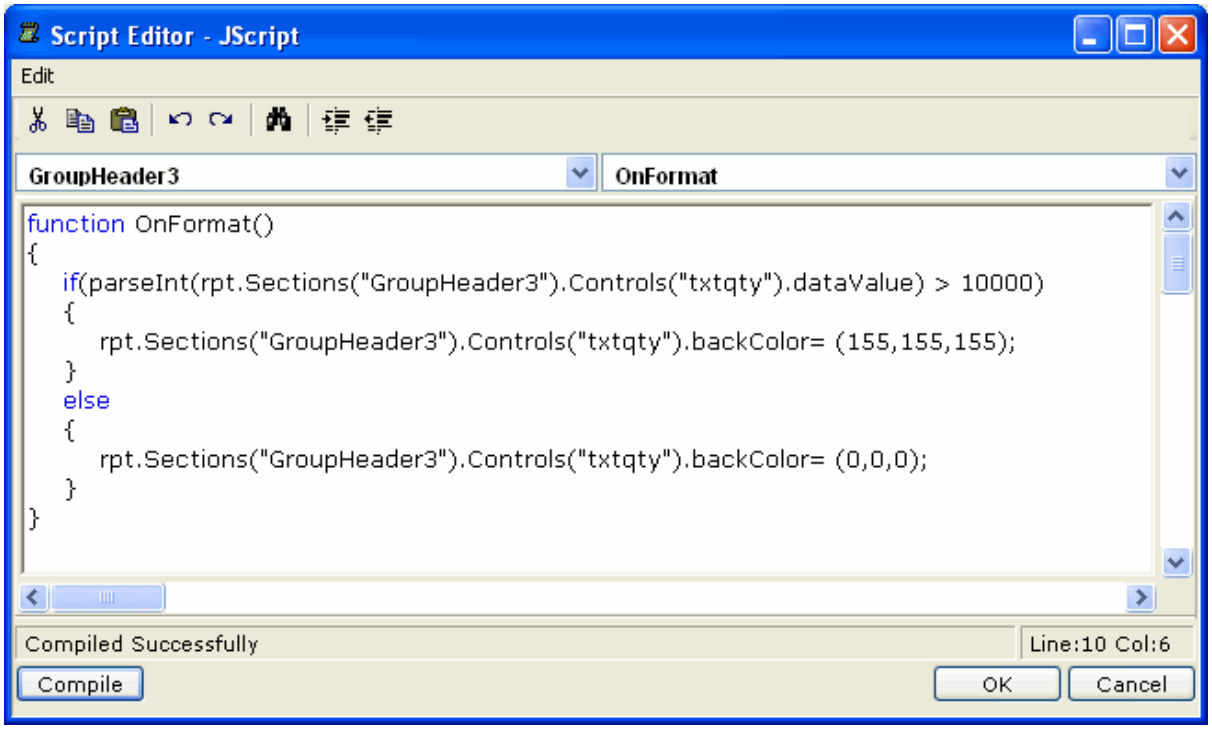

Figure 1: Script Editor

You can use the following objects and their properties to control the behavior of the report at runtime. These report objects are accessible in a specific hierarchy as explained below:

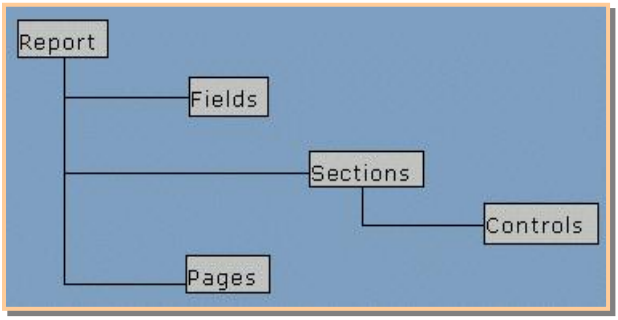

Figure 2: Report Object Hierarchy

The objects or events in the code editor are dependent on each other, for one object there is a specified set of events and vice-versa.

Context-Sensitive Help

The Script Editor also provides context-sensitive help that assists in correct code (syntax) formation. As you type the code in the Script Editor, the context-sensitive help keeps popping up selection list of various fields and objects that may fit into the syntax.

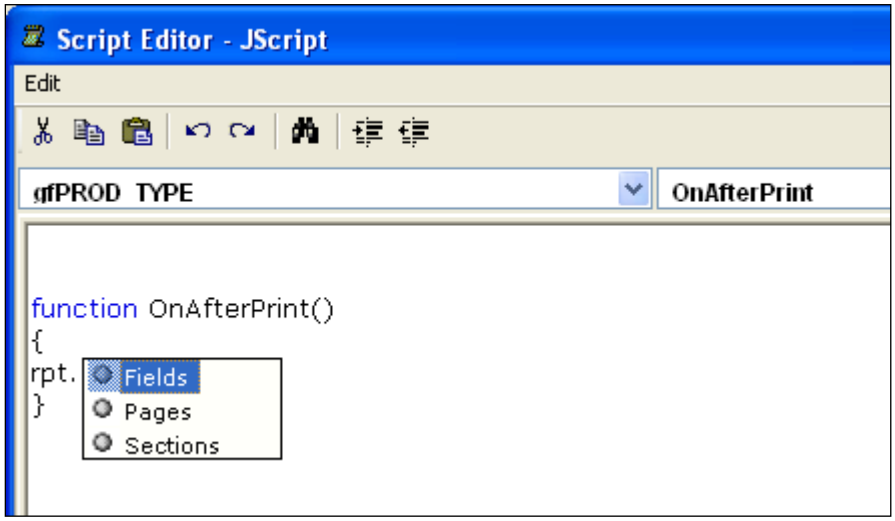

Figure 3: Context-Sensitive Help

Accessing Fields

You can access the report fields in 'rpt.Fields' collections. You can use this collection to write code in the Script Editor to access fields (controls) in the layout pane. Each event in the Script Editor has a specific purpose; you should not write a code that does not pertain to the object / event under which it has been written.

To add a new code under the Data Initialize event of the Intellicus Report Layout, the syntax is:

rpt.fields.add "<MyField>";

Warning: Make sure that the added field does not already exist; else, a fatal error will occur.

Code script for the value property of the fields can be given under 'OnFetchData' event of Intellicus Report Layout.

rpt.fields("SomeFieldName").value="<SomeValue>";

Accessing Layout Objects

The layout objects are the controls that are added to the report layout region. [See also: Working with Layout Editor, Chapter 3], you can access these objects through control's collection members of the sections collection.

rpt.sections("Detail").Controls("imgLogo").visible = false;

Important: You will not be able to access the database using code (scripts).

Conditional Formatting

You can achieve conditional formatting through Scripting too.

You can format a displayed row value if the values of that row satisfy a given condition. For example, if you need to compare the database field (say 'dbfield') with a field in the previous row, and encircle it in red if it is different, add a text box (say 'text box') in the report header and set its visible property (from the Properties list) as 'False'.

Set its text property (from the Properties list) to any arbitrary value (say '-999') that can never be attained by the field to be compared with.

Now, add a shape (say "shape1") around the 'dbfield' and set its color and shape to a red ellipse. The same can be achieved using the Script Editor as follows:

1. From the **Tools** menu, click **Scripting**.
2. Select object as *Detail*.
3. Select event as *OnFormat*.
4. Type the following JavaScript:

```
Object: Detail            Event: OnFormat

Code:
function OnFormat()
   {
     if ( ( rpt.Section("ReportHeader").Controls("txtBox").text != "_99999" ) &&
          ( rpt.Fields("dbValue").value !=
rpt.Sections("PageHeader").Controls("txtBox").text ) )
                {
                  rpt.Sections("Detail").Controls("Shape1").visible = true ;
                }
                else
                {
                  rpt.Sections("Detail").Controls("Shape1").visible = false ;
                }
      rpt.Sections("PageHeader").Controls("txtBox").text =
rpt.Fields("dbField").value;
   }
```

Conditional Suppressing Of Rows

You can suppress the display of certain rows as per your requirement, like some column containing NULL values can be suppressed (hidden) from getting displayed on the report.

There are two methods to do this:

- Select the control and set the visible property (Property window) value as false, and assign a 0 (Zero) value to the height property (Property window) of the control.
- Go to **Tools** > **Scripting**; select the object as *Detail*, and the event as *OnFormat*, and write the following code:

```
Object: Detail            Event: OnFormat
Code:
function OnFormat()
   {
     if ( rpt.Fields("Name").value == null )
     {
      rpt.Sections("Details").visible = false;
      }
     else
     {
       rpt.Sections("Details").visible = true;
     }
          if ( rpt.Fields("Name").value == null )
     {
      rpt.Sections("Details").height = 0;
      }
     else
     {
       rpt.sections("Details").height = 285;
     }
   }
```

Important: To dynamically change the height of a section through a program, the 'CanGrow' property (Properties list) of Detail Section should be set to 'False'.

If it is set to 'True', then the report section will override your (height) value to adjust the height of the section.

Conditional Calculation

You can calculate values in the report by giving conditions for calculation. For example, there are two fields in a report Account_type and Amount. There can be two account types say 'A' and 'B'. If you want to sum 'A' and 'B' separately, write a JavaScript in the Script Editor as:

```
Object: Report            Event: OnDataInitialize,
Code:
function OnDataInitialize()
{
  rpt.Fields.add("valueA");
  rpt.Fields.add("valueB");
}

function OnFetchData(eof)
{
             if ( rpt.Fields("STATUS").value == "A" )
             {
              rpt.Fields.("valueA").value = rpt.Fields("CUST_NO").value
              rpt.Fields(valueB").value = 0;
             }
             else
             {
               rpt.Fields("valueB").value = rpt.Fields("CUST_NO").value;
               rpt.Fields("valueA").value = 0 ;
             }
rpt.Sections(“Detail”).Controls(“extTYPE”).dataField=”valueA”;
rpt.Sections(“Detail”).Controls(“txtORDER_NO”).dataField = “valueB”;
}
```

This script will add two new fields in the report containing summated values for ‘A’ and ‘B’ account types.

Compiling scripts

After typing in the script, you can compile the script to make sure it will run without error and you will be able to achieve the results that you want, using the script. To compile the script, click the **Compile** button available on bottom-left side of the dialog box.

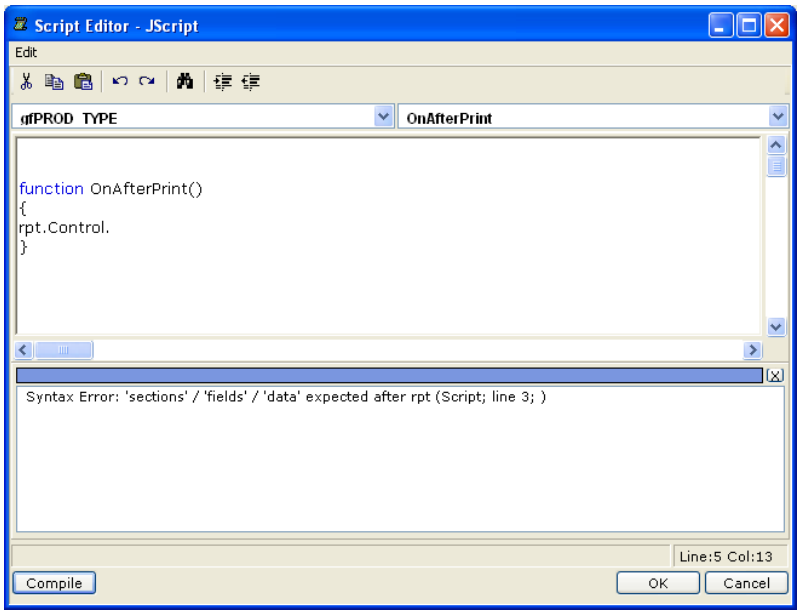

Figure 4: Syntax Error in the script

If the script has any syntax error, it is listed in a pane opening between script pane and buttons. You can remove the errors and click the **Compile** button to make sure the script is error-free.

Find and Replace

Script Editor dialog box offers Find and replace functionality. Click the Find button on the toolbar of Script Editor or press Ctrl + F on the keyboard to switch on the functionality.

You have options to **search up** and **search down**. Selecting **Match Case** will conduct a case sensitive search. Selecting **Match whole word only** will not find the words where the search string is part of a word. A click on **Find Next** button will start search.

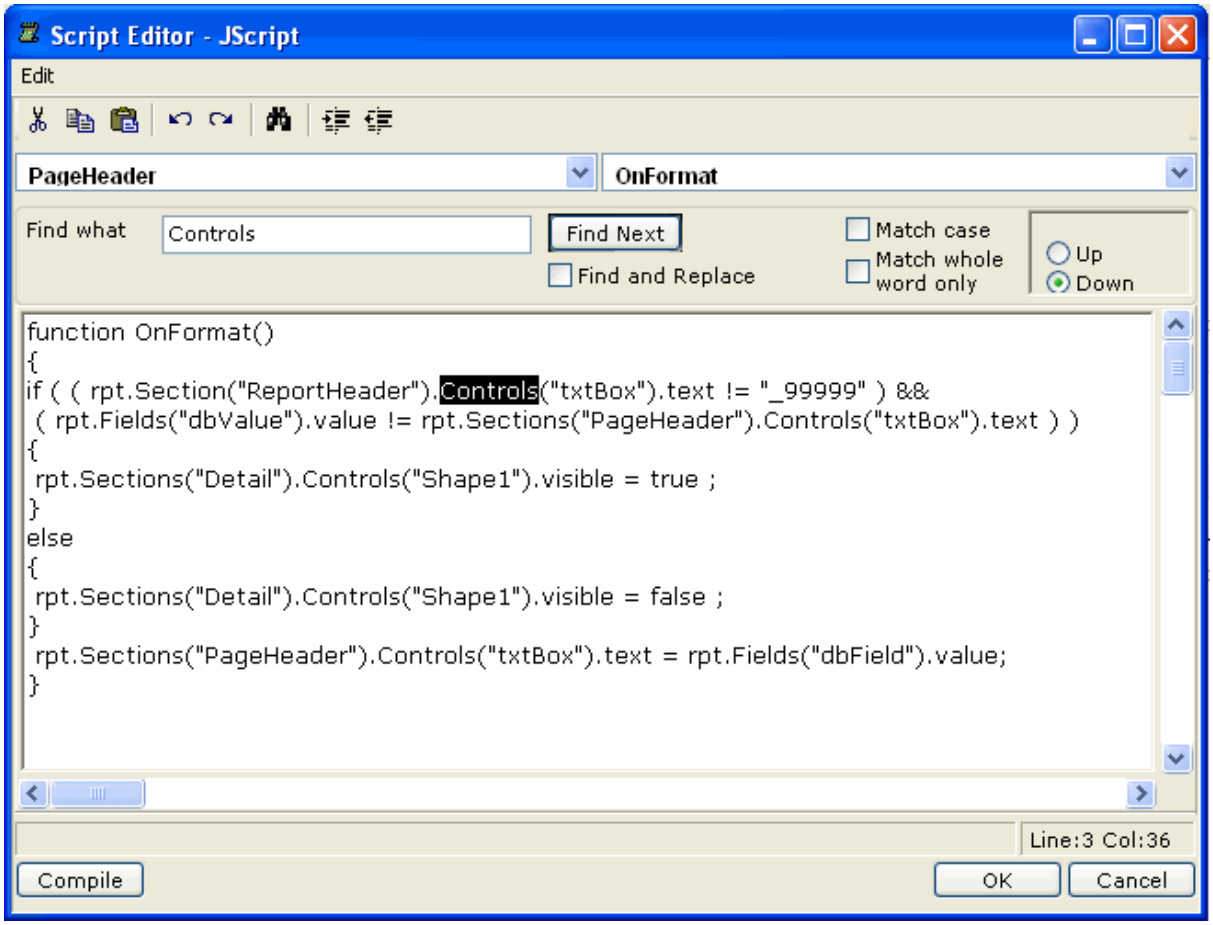

Figure 5: Find and Replace feature on Script Editor

If you want to carry out find and replace, select **Find and Replace** check box. Clicking **Replace** will replace the next occurrence of the search sting. Clicking **Replace All** will search for all the occurrences of the search string.