



## Report Logging for JavaScript Routines

---

This article describes how to enhance the built-in capabilities of the Tivoli Common Reporting tools using JavaScript. (Nov 2007)

---

From a forthcoming article to be published on [www.ibm.com/developerWorks](http://www.ibm.com/developerWorks)



IBM Tivoli is delivering the **Tivoli Common Reporting** software, aiming to provide customers with a reporting solution that is shared across the Tivoli portfolio and can be used by Tivoli, business partners, and customers as a base for increasing the value of solutions built around Tivoli products.

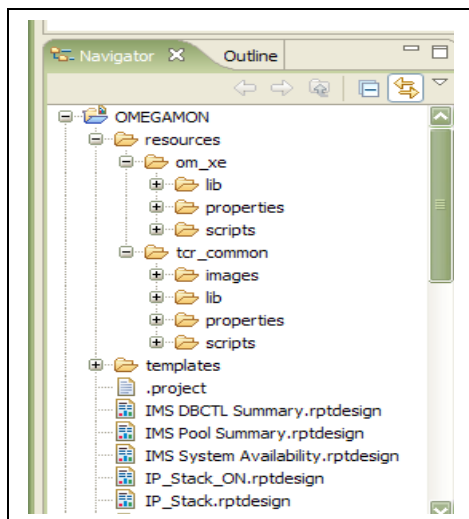
When developing new reports or modifying existing reports that use Tivoli Common Reporting and the Eclipse BIRT designer, it is often useful to enhance the built-in capabilities of the reporting tools using JavaScript. These JavaScript additions are often small, independent routines that are called as necessary when a report is being processed. BIRT provides a large number of places where JavaScript routines can be added to enhance the processing of the reports.

In development and often in production, it is useful to be able to trace the processing within these JavaScript routines in order to determine why a specific report is not performing as expected. Therefore, a report developer will often add specific trace messages or log messages to a report in order to diagnose where problems are occurring.

The Tivoli Common Reporting style package, supplied with Tivoli Common Reporting, provides an implementation of a logging method that can be used in both development and production environments. In a development environment, the logger writes messages for a report to a file specified within the report design. In production, the messages will be written to the Tivoli Common Reporting log file, and the logging of the messages can be turned on and off through the user interface.

### Connecting the logger script to your report

To use this routine with a report, you first ensure that the `Logger.js` script can be accessed by the report. Copy the `tcr_common` resource directory into the resources directory of your project. The following example shows the `tcr_common` directory copied into the `OMEGAMON` project :





The `tcr_common/scripts` folder contains collections of JavaScript functions, including the logger functionality, which can be used in report designs. These scripts can be called from the Expression Builder and used with almost any Text or Data item in a report. In addition, they can be called from customizable methods of the report, such as `beforeFactory()`, `beforeRender()`, or `beforeOpen()` of a data set.

To make the scripts available to a report, one or more **includeScripts** properties must be included in the report design file. These cannot be added using the Eclipse BIRT designer; you must add them manually. To do this, use the XML view of the Eclipse Report Design editor. The property elements should be added near the beginning of the `.rptdesign` file, before any parameter elements:

```
<list-property name="includeScripts">
  <property>tcr_common/scripts/ReportUtils.js</property>
  <property>tcr_common/scripts/DateTime.js</property>
  <property>tcr_common/scripts/Logger.js</property>
</list-property>
```

## Adding logging calls to the report

To initialize the logger within the report, you must add a few function calls to the initialize script of the report. Define a persistent variable named `logfileName` that contains the name of the file where the report log will be stored, then call the `setupLogging()` to set up the log file. In a report running within the BIRT designer, the log file will be created in the temp directory on the file system (`c:\temp` on Windows, `/tmp` on Unix or Linux platforms). To temporarily suspend logging in the designer, comment out the setting of the persistent variable. When running within the Tivoli Common Reporting server, the file in the temp directory will not be created; instead, the logging will be written to the standard Tivoli Common Reporting log location. The `logInitialize()` call places a short string in the log to indicate that logging has begun.

```
//
// Set up logging. Uncomment next line to enable in Designer tool.
reportContext.setPersistentGlobalVariable("logfileName",
"DateRangeParm.log");
setupLogging();
logInitialize();
// Add closeLogger(); to afterRender/afterFactory to close logger object.
```

To write a log or trace message, add a `debugLogger()` call that passes the string to log at the appropriate points in your code:

```
debugLogger("Customer Name=" + cust_name);
```

Finally, add a `closeLogger()` Javascript call into either the `afterFactory` or `afterRender` scripts. This call properly flushes and closes the log or trace data that has been written in the report.

```
debugLogger("we reached the afterFactory script");
// Close logger if it is enabled
closeLogger();
```

## Logging within common script functions

When writing common script routines that can be used by multiple reports, similar to the routines that are provided in the `tcr_common/scripts` directory, the code needs to ensure it works correctly in situations where the `Logger.js` script was not included in a report's definition. In this case, the code needs to make sure that the



debugLogger method exists before calling it; otherwise, a JavaScript error occurs. If the debugLogger function does not exist, the script code should either bypass the call or provide a substitute method. The following example defines a new method reference called dsDebuggerLogger; this reference points to debugLogger if it exists, or an empty function if it does not. The remaining code in the function uses the dsDebuggerLogger reference to write out the trace messages.

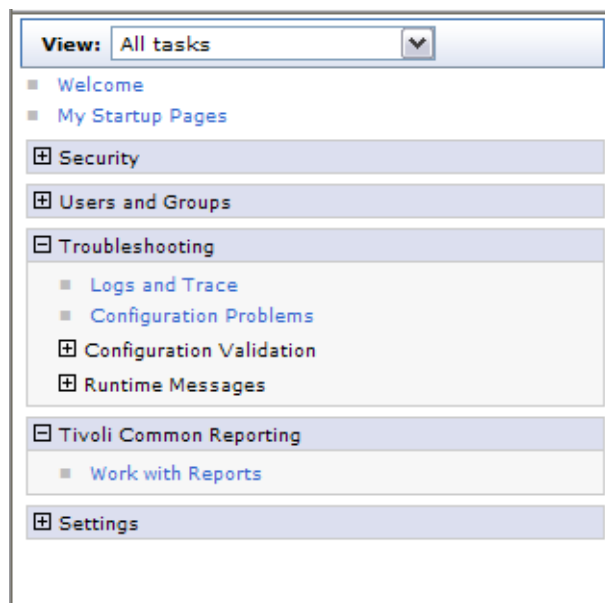
```
function ModifyQueryAddFilter (dataset,filter)
{
    dsDebuggerLogger = ( (""+typeof(debugLogger)) != "undefined" ) ? debugLogger :
    function(s) {};

    dsDebuggerLogger("queryText=" + dataset.queryText);
}
```

## Activating logging with the Tivoli Common Reporting server

To turn on or off logging within the Tivoli Common Reporting server, log in to the user interface as a user with administrator authority.

From the user interface, select **Troubleshooting->Logs and Trace** selection from the main navigation panel.



From the next panel, select the server to be modified. Typically, the server name is tcrServer. Select “Diagnostic Trace” and then select “Change Log Detail Levels.” The list of log attributes that are available for changing are displayed. Expand the com.ibm.tivoli.\* group to see the com.ibm.tivoli.reporting.reportLogger setting. Changing this setting will turn on or off logging using the Logger.js functions.



Logging and Tracing

Logging and Tracing > tcrServer > Diagnostic Trace Service > Change Log Detail Levels

Use log levels to control which events are processed by Java logging. Click Components to specify a log detail level for individual components, or click Groups to specify a log detail level for a predefined group of components. Click a component or group name to select a log detail level. Log detail levels are cumulative; a level near the top of the list includes all the subsequent levels.

Configuration Runtime

General Properties

Change Log Detail Levels

Components

Groups

\*=info

- com.ibm.iscportal.\*
- com.ibm.portal.\*
- com.ibm.sec.\*
- com.ibm.tivoli.\*
  - com.ibm.tivoli.reporting.\*
    - com.ibm.tivoli.reporting.audit.impl
    - com.ibm.tivoli.reporting.bulkImportExport.impl
    - com.ibm.tivoli.reporting.common.audit
    - com.ibm.tivoli.reporting.common.bulk
    - com.ibm.tivoli.reporting.common.dataAdministration
    - com.ibm.tivoli.reporting.common.datastore
    - com.ibm.tivoli.reporting.common.reportEngine
    - com.ibm.tivoli.reporting.common.runAndRender
    - com.ibm.tivoli.reporting.configuration
    - com.ibm.tivoli.reporting.datastore
    - com.ibm.tivoli.reporting.ejb
    - com.ibm.tivoli.reporting.plugin
    - com.ibm.tivoli.reporting.reportEngine.impl
    - com.ibm.tivoli.reporting.reportLogger
    - com.ibm.tivoli.reporting.reportViewer
    - com.ibm.tivoli.reporting.ui
    - com.ibm.tivoli.reporting.ui.json.command
    - com.ibm.tivoli.reporting.ui.json.controller
    - com.ibm.tivoli.reporting.ui.json.model
    - com.ibm.tivoli.reporting.util
- com.ibm.websphere.\*

## Sample code

The DateRangeParameters sample report, provided in the samples directory of the Tivoli Common Reporting style package, implements the logging function described within this article.



**Mack Phelps**  
 Advisory Software Engineer, IBM Tivoli Software  
[mphelps@us.ibm.com](mailto:mphelps@us.ibm.com)