Merative Social Program Management 8.1

Note

Before using this information and the product it supports, read the information in

Edition

This edition applies to Merative™ Social Program Management 8.0.0, 8.0.1, 8.0.2, 8.0.3, and 8.1.

© Merative US L.P. 2012, 2023

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.

Contents

1 Troubleshooting a problem

Troubleshooting is a systematic approach to solving a problem. The goal of troubleshooting is to determine why something does not work as expected and how to resolve the problem.

The first step in the troubleshooting process is to describe the problem completely. The following is a list of basic questions to help you identify and describe the problem:

- What are the symptoms of the problem?
- Who, or what, is reporting the problem?
- What are the error codes and messages?
- How does the system fail? For example, is it a loop, hang, crash, performance degradation, or incorrect result?
- Where does the problem occur?
- When does the problem occur?
- Under which conditions does the problem occur?
- Can the problem be reproduced?

The answers to these questions typically lead to a good description of the problem, which can then lead you a problem resolution.

Two main ways to approach any problem you encounter are using log files and understanding messages.

1.1 Examine the log files

Examine log fails to find out what happened and what caused the error.

Merative™ Social Program Management is a part of a complex, multitier environment. Information about different parts of the system can be found in a number of different logging sources. You should be aware of these other log sources when you are troubleshooting a problem. Refer to the documentation for other products relevant to your environment for information on the log sources they produce and how to get them.

Logging is provided by the *curam.util.resources.Trace* class that provides a wrapper for the Apache Log4j 2 API. The final destination of the trace information is configurable and can be set as a log file associated with the application server, a standalone log file, a console, or a database.

1.2 Where does the problem occur?

Determining where the problem originates is not always easy, but it is one of the most important steps in resolving a problem. Many layers of technology can exist between the reporting and failing components.

Most errors occur in the application server or database environments. However, there are many components involved in a Social Program Management environment, so be aware of interactions with networks, I/O systems, and so on. The following questions help you to focus on where the problem occurs to isolate the problem layer:

- Is the problem specific to one platform or operating system, or is it common across multiple platforms or operating systems?
- Is the current environment and configuration supported?

If one layer reports the problem, the problem does not necessarily originate in that layer. Part of identifying where a problem originates is understanding the environment in which it exists. Take some time to completely describe the problem environment, including the operating system and version, all corresponding software and versions, and hardware information. Confirm that you are running within an environment that is a supported configuration. Many problems can be traced back to incompatible levels of software that are not intended to run together or have not been fully tested together.

Related information

[Social Program Management Prerequisites](#)

1.3 When does the problem occur?

Develop a detailed timeline of events that lead up to a failure, especially for those cases that are one-time occurrences.

When reviewing logs, start with the time the error was first reported or identified and try to identify the last point when the system was working normally. Then, begin looking at the initial error or errors. Typically, errors can be cascading and the most recent information can be far removed from the initial issue.

To develop a detailed timeline of events, answer these questions:

- Does the problem happen only at a certain time of day or night?
- How often does the problem happen?
- What sequence of events leads up to the time that the problem is reported?
- Does the problem happen after an environment change, such as upgrading or installing software or hardware?

Responding to these types of questions can give you a frame of reference with which to investigate the problem.

1.4 Under what conditions does the problem occur?

Knowing which systems and applications are running at the time that a problem occurs is an important part of troubleshooting.

These questions about your environment can help you to identify the root cause of the problem:

- Does the problem always occur when the same task is being performed?
- Does a certain sequence of events need to occur for the problem to surface?
- Do any other applications fail at the same time?

Answering these types of questions can help you explain the environment in which the problem occurs and correlate any dependencies. Remember that just because multiple problems might have occurred around the same time, the problems are not necessarily related.

1.5 Can the problem be reproduced?

From a troubleshooting standpoint, the ideal problem is one that can be reproduced.

Typically, when a problem can be reproduced you have a larger set of tools or procedures at your disposal to help you investigate. Consequently, problems that you can reproduce are often easier to debug and solve. However, problems that you can reproduce can have a disadvantage: If the problem is of significant business impact, you do not want it to recur. If possible, re-create the problem in a test or development environment, which typically offers you more flexibility and control during your investigation.

- Can the problem be re-created on a test system?
- Are multiple users or applications encountering the same type of problem?
- Can the problem be re-created by running a single command, a set of commands, or a particular application?

2 Gathering information about a problem

Describes the type of information about a problem you should gather when you contact Social Program Management support.

Problem descriptions help you and the Social Program Management Support representative know where to start to find the cause of the problem. It is important to gather as much information as possible about the circumstances of the issue.

To help you with detailed must-gather information for different product areas, see the Collecting data for Merative™ Social Program Management issues support article.

The following list describes the minimum set of information and files to have at hand when you contact customer support:

• The affected version of Social Program Management.
• The versions of third-party tools that are used with Social Program Management, for example, your application server, database, operating system, Java version.
• A clear description of the problem.
• Error logs, where applicable.
• Relevant files, where applicable

Further details about a problem to be considered:

• Was the function previously working? If so, provide details about the version of Social Program Management where the function was previously working.
• What are the exact components installed, including all solution modules?
• What type and version of application server is the application running on?
• A detailed description of the deployment architecture may be required. For example, is the application server in a clustered environment?
• Is this Social Program Management application a new instance or migrated instance of the application?
• In what operating system and version is the problem encountered?
• In what browser type and version is the problem encountered?
• What, if any, globalization settings are configured?

2.1 Development environment configuration settings

The configuration of your development environment can often be the source of problems. The **configreport** build target tells you what settings and property files are in your system.

Run **appbuild configreport** from a command prompt opened at *%CURAM_DIR% \EJBServer*.

%CURAM_DIR% is the Social Program Management installation directory, which by default is *C: \Merative\Curam\Development*.

appbuild configreport creates a *%CURAM_DIR%\EJBServer \config_report.zip* file containing all the relevant settings and software versions on the machine. This is useful if remote support is required to analyze environment/configuration issues.

The information gathered includes environment settings for Social Program Management specific and system environment variables, installer logs, and so on.

> **Note:**
>
> The -v parameter for a build target provides verbose tracing. To print the verbose output to a log file, run the following:`build server -v > trace_file.txt`.

2.2 Application server configuration and deployment settings

The *appserver.properties*, *deployment_packaging.xml*, and *application.prx* files contain useful information about issues that relate to application server configuration and deployment.

- *%CURAM_DIR%\EJBServer\project\properties\AppServer.properties*. This file used by the configure build target. *%CURAM_DIR%* is the Social Program Management installation directory, which by default is *C:\Merative\Curam \Development*.
- *SERVER_DIR\project\config\deployment_packaging.xml*. This file is used when building the application EAR to split the client components into WAR and EAR files. *deployment_packaging.xml* also allows finer control of the EAR configuration and included modules.
- *%CURAM_DIR%\EJBServer\project\properties\Application.prx*. The properties contained in this file are loaded into the database when the build database target is run. The properties are cached from the database for use by the application at runtime. The runtime properties, which can be altered by the system administration application, are available via Social Program Management JMX statistics. For issues relating to the runtime configuration of the application server you can use the application server administrative console to compare your configuration against a default deployment.

To get a static copy of the application server configuration, use the tool that is appropriate for your application server:

- You can use IBM® Support Assistant Data Collector (ISADC) for all versions of WebSphere® Application Server.
- For Oracle WebLogic Server, consult the WebLogic Server documentation on how to get archives of the configuration.
- `Kubernetes` For WebSphere® Application Server Liberty, use the `$WLP_HOME/bin/server package` command.

When providing configuration information to Merative™ Social Program Management support, the files that the server configuration tool creates must be included in your submission.

2.3 Application server logs

The application server logs are the first place that you look when you get an exception or error in a deployed application. Logs can optionally be set to verbose to provide additional information about the running application server.

Application server logs

- **IBM® WebSphere® Application Server application logs**

  WebSphere® Application Server application log files can be found in the logs folder for each profile in the cell; for example, `$WAS_HOME/profiles/<profile_name>/logs`. There is a folder for each server in the logs directory. This location is where the JVM stdout and stderr logs (`SystemOut.log` and `SystemErr.log`) are stored. However, you can specify alternative locations and names for these files. Depending on the log settings, multiple files can be created.

- Kubernetes **WebSphere® Application Server Liberty application logs**
  By default the WebSphere® Liberty logs are stored in `server.config.dir/logs/`, where *server.config.dir* is the property that represents a server-specific configuration directory.

  Include more detailed information about console.log (stdout & stderr), messages.log (similar to console.log with timestamps, but no direct JVM log messages), and trace.log if you want.

- **FFDC logs**

  FFDC logs provide essential information about a running application server. FFDC logs for individual profiles log all the errors that are encountered during the profile runtime. FFDC logs are in `logs/ffdc`. The first time an error is encountered during the life of a WebSphere process a set of FFDC files are created, which have more detail than the JVM logs.

  FFDC log files for individual profiles log all the errors that are encountered during the profile runtime. These files are in the `logs/ffdc` directory. When you provide information to Social Program Management support, create a compressed file of the logs folder for all profiles in the cell.

- **Oracle WebLogic Server logs**

  WebLogic Server logs can be found in `%WLS_HOME%\user_projects\domains\<domain_name>\servers\<server_name>\logs`. `<servername>_redirect.log` is useful. Ensure that the *WLS_REDIRECT_LOG* environment variable is set to the correct location for the log file. When you provide information to a support representative, create a compressed file of the logs folder for all servers in the configuration.

Social Program Management JMX statistics

JMX statistics can be downloaded as a compressed XML file by accessing the URL `/Curam/JMXStats.do` or `/Curam/JMXStats.do?action=download`. Provide Social Program Management JMX statistics to a support representative when you are troubleshooting a performance or performance-related issue.

2.4 XML server logs

Consult the XML server log if you encounter problems with the XML server or XSL templates and PDF, RTF, or plain text rendering.

The XML server logs its output to the *XMLServer.log* file in the *\XMLServer* directory. This log is not controlled by application properties as it does not require a connection to the database.

2.5 Application server version

The application server version helps your customer representative to diagnose your problem.

The application server logs contain the version of the application server.

You can also enter the following at the command line to determine the version of your application server.

- IBM® WebSphere® Application Server: **%WAS_HOME%/AppServer1/bin/ versionInfo.bat**
- Oracle WebLogic Server: In *WLS_HOME/bin*, run **setDomain.[cmd][sh]**, and then **java weblogic.version**.
- Kubernetes WebSphere® Application Server Liberty: **$WLP_HOME/bin/productInfo version**

2.6 Operating system version

The operating system version will aid your customer representative in diagnosing your problem.

On Microsoft™ Windows™, enter **winver** at the command prompt.

On Linux®, enter **uname -a** on the command line.

2.7 Java™ version

The Java™ version will aid your customer representative in diagnosing your problem.

To find the version of Java™ you are using, enter **java -version** at the command line.

Many versions of Java™ can be installed on a platform. To find out the specific Java™ version that you are using, ensure that the PATH environment variable or Java install location is set correctly. The application server logs will usually include the Java™ version that's specific to the application server, which can vary across nodes.

2.8 Database version

The database version will aid your customer representative in diagnosing your problem.

For IBM® Db2®, **db2level** prints the version and service level of the installed DB2 product to the console by default.

For Oracle Database, connect to the database using the **sqlplus** command and log in to see the version.

2.9 Database connectivity

Use the **configreport** build target to troubleshoot issues with your database connection.

Run **appbuild configreport** from a command prompt opened at *%CURAM_DIR% \EJBServer* to produce a *config_report.zip* file in *%CURAM_DIR%\EJBServer*. *%CURAM_DIR%* is the Social Program Management installation directory, which by default is *C: \Merative\Curam\Development*.

The configreport output has a section called "-- DATABASE CONNECTIVITY --." that reports if a successful connection to the database has been made.

Database connection settings

Bootstrap.properties defines how the connection to the database is made. Start with this file when you are troubleshooting a database connection.

- *%CURAM_DIR%\EJBServer\project\properties\bootstrap.properties* The *bootstrap.properties* file mainly contains the minimum set of properties necessary for obtaining a connection to the database.
- The *properties.jar* file that is located in the application .EAR contains the *bootstrap.properties* file which is used by the deployed application at run time.

3 Troubleshooting a blank or frozen page

Intermittent blank or frozen application pages can sometimes be caused by a combination of performance-related issues. Performance issues are often related to a combination of user hardware specification, browser choice, network bandwidth, web server, or Merative™ Social Program Management application server configuration.

Investigate the following areas when a blank or frozen page is encountered.

3.1 Network, bandwidth, and server performance

Heavy network load and bandwidth usage might cause blank or unresponsive pages.

Review the network and bandwidth load at the time of blank or unresponsive page reports.

Review the load on the web server and Merative™ Social Program Management application server. A heavy load on the web server or application server can cause a slow response or timeout issue at the browser.

Gather information for performance analysis, see Collecting data: Questions for performance issues.

Contact support for specific tuning recommendations that can improve performance.

3.2 Internet browser and hardware specification

Merative™ Social Program Management supports multiple browsers, but not all browsers are the same. The choice of internet browser can impact on page response times.

For information about software requirements including browser support, see the Merative™ Social Program Management Prerequisites.

For information about hardware requirements, see Hardware requirements for development and testing.

3.3 Persistent connection settings

The settings that are associated with persistent connections may contribute to blank Social Program Management application pages.

HTTP persistent connection, or HTTP keep-alive, is where a single connection is used to send and receive multiple HTTP requests. Persistent connections can be closed by the server or the browser based on idle timeout values. The timeout settings for keep-alive connections vary between browsers, applications servers, and web servers; and can be configured. When intermittent blank screens are encountered, you should review the keep-alive connection timeout values.

When a persistent connection is closed due to a timeout, the browser detects the closing state and might repost the request.

3.4 User perception

Slower than usual rendering times can cause the user to perceive a blank page has been displayed.

When a page is slow to load, impatience can cause the user to restart or refresh the browser repeatedly before the page loads. In addition, in this situation users sometimes click around the browser in an attempt to get it to respond. The latter response can cause the browser to freeze by opening too many connections with the server. When investigating blank screen reports, user usage scenarios should be captured to understand if the blank screen reports are a result of user frustruation with a slow response times.

3.5 Pages with long lists

Performance issues can be experienced when rendering page content that contain long lists.

Long list pages in Merative™ Social Program Management that contain hundreds or even thousands of items, will be slow to display in the browser. For such pages, it is recommended to make use of pagination support and design the page to include a search function to narrow the results. See the related reference for information on how to handle long lists.

3.6 Reproducible JavaScript™ errors

If the same usage scenario consistently reproduces a blank page, an underlying JavaScript™ error is potentially the root cause.

A blank page that is encountered in the same reproducible scenario can usually be attributed to an underlying JavaScript error that prevents the Merative™ Social Program Management loading mask from being removed. This situation is identified by a reproducible scenario, even if intermittent, in addition to the existence of a JavaScript error displayed in the browser console.

To identify the JavaScript error, inspect the browser developers tools console by pressing F12 in the browser. You should also use the developer tools to inspect the HTML content for the blank screen. The HTML content may be available, but hidden because of CSS (when *visibility:hidden* is set). When such issues are encountered, contact Merative™ Social Program Management support.

3.7 Gathering information about a blank or frozen page

When you contact support about problems that are related to blank or frozen pages, gather information that helps your support representative diagnose the problem.

Browser version and compatibility mode settings

Provide full details of the browser version that the issue is encountered on. Check if the issue is reproducible on other supported browsers.

Machine specification

Provide details of the machine operating system, specification, and available memory for users when the problem is encountered.

Usage scenario

Provide details of the usage scenarios that help narrow down the cause of the problem. If the issue is intermittent, usage scenarios are still helpful. Also include the number of tabs open when the problem is encountered.

User recovery

Provide details of how a user recovers from the issue. For example, does the browser crash and need to be restarted? Does the user refresh the page and start again?

Server response times

Record the following information, if applicable.

- Are there performance issues with the server when the error is reported?
- Are there any performance concerns in general?
- Does the user encounter the issue in the local office or are they in a remote location?
- What bandwidth is available to the users in remote locations?
- Does the problem occur more frequently at certain times of the day?

System configuration details

Record the following information, if applicable.

- Is the system in a clustered environment? If so, how many clusters and servers are available?
- How many concurrent users access the system?
- Is the static content server feature in use? If so, how is it configured and what HTTP response headers are set?
- What is the HTTP keep-alive timeout value set to?

HTML source for the blank page

Provide the HTML source for the blank page encountered. In Chrome, ensure that the frame source is provided.

JavaScript errors

Provide details and screen captures of any JavaScript errors in the browser. Open the console window to confirm any reported errors.

Browser JavaScript logs

Where possible, enable the client and browser tier debugging and gather the log files when the issue is encountered. To enable logging, set the following properties in `curam/omega3/i18n/CDEJResources.properties`:

- TraceOn=true
- JavaScriptTraceOn=true

This file can be added to your webclient project in a custom component before you build the application. At run time, a copy of the file can be found in properties.jar located in the *WEB-INF \lib* directory of the *ClientModule.war* file (located in the Social Program Management EAR).

For example, in IBM® WebSphere® Application Server:

1. Locate the client *WEB-INF\lib* directory: *<WAS_HOME>profiles\<profile> \installedApps\<node>\Curam.ear\ClientModule.war\WEB-INF\lib*.
2. Extract a copy of CDEJResource.properties from the properties.jar located in this directory. The file should have the package curam\omega3\i18n.
3. Place this copy in *<WAS_HOME>profiles\<profile>\installedApps\<node> \Curam.ear\ClientModule.war\WEB-INF\classes\curam\omega3\i18n*.
4. Modify and set the properties as required.
5. Restart the server to apply the changes.

These changes may have system performance implications. It might not be possible to set the changes in a production system.

Local JavaScript tracing in the browser

As an alternative to JavaScript™ logs, you can set the browser JavaScript™ tracing locally for a user with a web debugging proxy tool such as Fiddler Web debugger. While using the web debugging proxy tool, set the DEBUG JavaScript™ variable to true.

Modifying the DEBUG Variable in Fiddler

1. Select **Rules** > **Customize Rules** in the Fiddler menu.
2. In the file that opens, find the onBeforeResponse function and add the following line: `static function OnBeforeResponse(oSession: Session) {`
   ...... oSession.utilReplaceInResponse("var DEBUG=false;", "var DEBUG=true;");}
3. Save the file.

Now, when you access the application with Fiddler running, debug output is recorded. In addition to the browser console log, you can use the traffic that Fiddler captures for debugging. The captured traffic shows the network traffic, and whether connections are being lost or dropped.

> **Note:** You need to decrypt the HTTPS content in fiddler. This can be done using the **Tools** > **Fiddler Options** menu. Select the HTTPS tab and tick the decrypt HTTPS traffic option.

Related information
[Fiddler Web Debugger](#)

4 Resolver page impact caused by the "double load" solution

Condition

When a submit action is triggered from a modal dialog and that dialog must be closed after the server response, a double load/request occurs. The double load is a possible performance issue and also does not deliver the ultimate user experience

Cause

UIM pages that contain the *JSP_SCRIPLET* element are normally used for redirecting users to other pages of the application based on the logic that is contained within that *JSP_SCRIPLET* element. Such UIM pages are called "resolve" pages and this change affects them under the following usage patterns:

• To address the double load issue, post submit processing logic verifies whether the request is coming from a modal dialog and also if it redirects to another page. This logic prevents the redirection to the new page and takes the responsibility of manipulating the *HttpServletResponse* object to be returned.
• The **Post Process Response** page is sent only on form submissions from modal dialogs and when the next page is different than the current page.
• The "double load" solution is delivered as part of the infrastructure. However, some resolve pages might contain *JSP_SCRIPTLET* elements that manipulate the screen context or use JavaScript to close and refresh the parent pages. Those resolve pages could be impacted by this change.
• If you have resolve pages that exhibit the usage patterns that are described, the following sections show how to address those impacts:

Remedy

Procedure

UIM pages that contain the *JSP_SCRIPLET* element are normally used for redirecting users to other pages of the application based on the logic that is contained within that *JSP_SCRIPLET* element. Such UIM pages are called "resolve" pages and this change affects them under the following usage patterns:

1. You have resolve pages that specify a *JSP_SCRIPLET* element that changes the RPU (Return Page URL) and changes the Screen Context to 256 (Modal).
2. You have resolve pages that specify a *JSP_SCRIPLET* element that contains JavaScript to close and refresh the parent page from which that resolve page was started.

4.1 Changing the Return Page URL and the screen context to 256 (modal)

In this case, the resolve page redirects to the parent tab page by using the Return Page URL (RPU) and changes the Screen Context to 256 (Modal). The screen context change causes unexpected behaviors such as the controls that appear on the bottom of the page instead of in the tab's menu as expected. That happens because the page is loaded as if the content is within a modal rather than a tab.

To address this issue in affected resolve pages, update the pages by removing the screen context parameter as in the following example.

Look for:

```
<JSP_SCRIPTLET>
    …
    curam.omega3.request.RequestHandler
      rh = curam.omega3.request.
        RequestHandlerFactory.getRequestHandler(request);

    url += "?o3ctx=256";
    response.sendRedirect(response.encodeRedirectURL(url));
  </JSP_SCRIPTLET>
```

Change to:

```
<JSP_SCRIPTLET>
    …
    curam.omega3.request.RequestHandler
      rh = curam.omega3.request. \\
        RequestHandlerFactory.getRequestHandler(request);

    url += += "?" + rh.getSystemParameters(); \\
    response.sendRedirect(response.encodeRedirectURL(url));
  </JSP_SCRIPTLET>
```

4.2 Closing the modal dialog and refreshing the parent page and modal dialog

Scenarios may also exist where a modal dialog is launched by an item from a list loaded after submitting a search.

An example of this dialog is when you are searching for a person, the search provides you with a list of people where every entry in the list has an Edit button that opens a modal. When that edit modal dialog is closed clicking on the submit button, the user is redirected back to the original search page. The original search criteria is persisted on the search page and the list is populated using that search criteria again. Currently this behavior is achieved with a pattern which uses the resolve page that receives the fields used to filter the results and outputs HTML code that, via JavaScript, which performs the following actions

- Accesses and submits the parent page's form.
- Closes the modal dialog.

Applying the changes to address this double load scenario, this pattern is impacted because the resolve page is being loaded directly on the parent page instead of the modal. Hence the parent

page is no longer accessible and cannot submit the form via JavaScript. The impact on this pattern may be addressed by setting the modal's action link property *DISMISS_MODAL* to false in all the UIM pages that link to these types of resolve pages. This avoids the scenario where the required modal dialog is closed by the new post processing logic.

Look for the resolve page containing the described JavaScript pattern

Look for:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PAGE
  PAGE_ID="TestResolve_resolveCloseModal"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file://Curam/UIMSchema.xsd">
<JSP_SCRIPTLET>
      …

    out.println("<script type='text/javascript'>");
    out.println("require(['dojo/ready', 'curam/core-uim', 'curam/util/Dialog'],
 function(ready) {");
    out.println("curam.util.Dialog.init();");
    out.println("ready(function() {");
    out.println("curam.util.Dialog.pageLoadFinished();");
    out.println("curam.util.Dialog.closeAndSubmitParent();");
    out.println("});");
    out.println("});");
    out.println("</script>");
 </JSP_SCRIPTLET>
...
</PAGE>
```

Look for the UIM with the link to the resolve page:

```xml
<PAGE
  PAGE_ID="TestResolve_modifyPerson"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file://Curam/UIMSchema.xsd">
  ...

  <ACTION_CONTROL
      IMAGE="SaveButton"
      LABEL="ActionControl.Label.Save.Resolve.Close.Modal"
      TYPE="SUBMIT">
      <LINK PAGE_ID="TestResolve_resolveCloseModal">
          <CONNECT>
           <SOURCE
              NAME="PAGE"
              PROPERTY="id"
           />
           <TARGET
              NAME="PAGE"
              PROPERTY="id"
           />
        </CONNECT>
        </LINK>
    </ACTION_CONTROL>
  ...
</PAGE>
```

Change the dialog by inserting the DISMISS_MODAL="false" attribute on the link

```
<PAGE
  PAGE_ID="TestResolve_modifyPerson"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="file://Curam/UIMSchema.xsd">
  ...

  <ACTION_CONTROL
      IMAGE="SaveButton"
      LABEL="ActionControl.Label.Save.Resolve.Close.Modal"
      TYPE="SUBMIT">
      <LINK PAGE_ID="TestResolve_resolveCloseModal" DISMISS_MODAL="false">
          <CONNECT>
           <SOURCE
             NAME="PAGE"
             PROPERTY="id"
           />
           <TARGET
             NAME="PAGE"
             PROPERTY="id"
           />
          </CONNECT>
         </LINK>
      </ACTION_CONTROL>
  ...
</PAGE>
```

5 Troubleshooting batch processes

Condition

Always check the *BatchLauncher<DATE_TIME>.log* file for errors and check the
<BatchName>_<DateTime>.log file for a count of skipped cases.

Remedy

Procedure

1. Depending your deployment topology, you might have to review multiple locations to view
   the error log.
2. If you run the stream component of the batch on multiple servers, you will need to review
   each servers' *BatchLauncher<DATE_TIME>.log* file for errors. The BatchLauncher
   will contain useful stack traces in the event of technical errors.
3. In addition to the BatchLauncher, the exception details will be captured in the database for
   technical errors.

6 Contacting Merative ™ Social Program Management Support

If you still need help after trying to find a solution by using self-help options such as product documentation and support articles, contact [Merative ™ Social Program Management Support](#).

7 Requesting a product enhancement

You can request a product enhancement for Social Program Management at the [Merative ™ Ideas Portal](#)

8 Known limitations

A list of known limitations for Merative™ Social Program Management or supported related products. Workaround information is provided where available.

8.1 Empty tab container on page

Condition

An empty tab container appears on a page with conditional links.

Cause

When in page navigation contains conditional links, if none of the conditional links are displayed, an empty tab container is drawn on the page.

8.2 Errors in Dynamic UIM modal dialog wizard pages

Condition

Linked modal dialog pages can sometimes have errors that are caused by duplicate page-ids.

Cause

When Dynamic UIM modal dialog wizard pages are linked with an explicit link in the ACTION_CONTROL label, page display errors can be caused by pages that do not have distinct page-ids, for example, if one page id is a substring of the other.

8.3 File download error reporting

Condition

A file download that does not complete successfully does not report an error.

Cause

A file download that is started from a tab level action menu that fails to complete does not report an error.

Remedy

Procedure

1. To troubleshoot a file download failure, enable client logging and search the Eclipse console for errors. Alternatively, use an HTTP monitor (for example, Firebug NET panel) and search for failed requests to `servlet/FileDownload`.

2. Examine the response for more error information.

8.4 Error displayed in the application server's default error page

Condition

Errors are displayed in the application server's default error page instead of the Cúram error page when the message does not exist in the catalog.

Cause

When the application attempts to report a message from a server message catalog, and that message does not exist in the catalog; the message not found error is displayed in the application server's default error page instead of the Cúram error page.

8.5 Action sets not displayed in modal dialog page

Condition

Page level action sets are not displayed in a modal dialog that contains an SVG tree widget.

Cause

A modal dialog that contains an SVG tree widget does not display any page level action sets.

Remedy

Procedure

Define the action set within a cluster.

8.6 Runtime error displayed when a string literal is not properly closed by a double quotation mark

Condition

A runtime error can be caused by a new line escape character in properties files.

Cause

UIM properties files do not support the use of new line escape sequences (\n) to alter the layout of screen content such as page descriptions or field labels. A runtime exception error message is displayed: `String literal is not properly closed by a double quote.`

Remedy

Procedure

To prevent the error, remove the escape sequence from the properties file.

8.7 JavaScript error in the rules decision tree widget

Condition

New line characters can cause a JavaScript error in the rules decision tree widget.

Cause

The rules decision tree widget does not support new line characters in the text that is displayed in nodes and for input data. These characters can cause a JavaScript error that results in the widget not being displayed.

Remedy

Procedure

Ensure that new line characters are not used in the rules decision tree widget.

8.8 Field level validation messages

Condition

Field level validation messages do not appear in the same order as the fields are displayed.

Cause

If a user does not input information in the mandatory fields in the Cúram application, the field level validation messages do not appear in the same order as the fields are displayed.

8.9 RuleObjectPropagatorConfigurationSnapshot classes are not for external use

Condition

Access to the `RuleObjectPropagatorConfigurationSnapshot` classes is restricted. The classes are not intended for external use. Incorrect use of the `RuleObjectPropagatorConfigurationSnapshot` classes might cause performance issues.

Cause

Incorrect use of the `RuleObjectPropagatorConfigurationSnapshot` classes, might cause performance issues and, in some cases, might cause a system outage.

Remedy

Procedure

The `RuleObjectPropagatorConfigurationSnapshot` classes that are in the *curam.core.sl.infrastructure.propagator.impl* package are not intended for external use. If you are using any of the methods of the `RuleObjectPropagatorConfigurationSnapshot` classes, contact support for further assistance.

Related tasks

[Contacting Merative Social Program Management Support on page 29](#)

8.10 The CER Editor strips comments from XML in a rule set

The CER Editor strips comments from XML in a rule set.

Condition

When you save a rule set in the CER Editor, comments that have been added externally using an XML Editor are erased automatically.

If you use an XML editor to change a rule set and to add comments, when the rule set is re-opened in the CER Editor and then saved, the comments are removed. This is a limitation because of Adobe Flex. Potential solutions will negatively impact the performance of the CER Editor when opening larger rule sets.

Remedy

Procedure

To mitigate against comments being erased, the CER Language does support comments. All elements support the concept of annotations. An example of this is shown below:

```
<compare comparison="&lt;">
   <Annotations>
     <Documentation text="This comment demonstrates how a comment can be added to a CER
 Element.
                      In this example, the element is a compare." />
   </Annotations>
   <Number value="0" />
   <Number value="1" />
</compare>
```

8.11 Diagram disappears after export of many diagrams in a large ruleset

This issue applies to the stand-alone CER Editor.

Condition

A diagram disappears after the export of a large number of diagrams in a large rule set.

Remedy

Procedure

Click another diagram and then return to the diagram to make the diagram visible.

8.12 Progress bar doesn't update during export of diagrams in the Cúram Express Rules Editor

This issue applies to the stand-alone CER Editor.

Condition

When you export a large number of diagrams in a large rule set using the standalone CER Editor, the progress bar does not provide information on the progress of the export.

Remedy

Procedure

As the process is running, a spinner is displayed that indicates that the CER editor is busy. When the export is complete, the progress bar shows that the process is complete and the you are prompted to save the export.

Notices

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the Merative website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

Information concerning non-Merative products was obtained from the suppliers of those products, their published announcements or other publicly available sources. Merative has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-Merative products. Questions on the capabilities of non-Merative products should be addressed to the suppliers of those products.

Any references in this information to non-Merative websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this Merative product and use of those websites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to Merative, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy

The Merative privacy policy is available at https://www.merative.com/privacy.

Trademarks

Merative ™ and the Merative ™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Adobe™, the Adobe™ logo, PostScript™, and the PostScript™ logo are either registered trademarks or trademarks of Adobe™ Systems Incorporated in the United States, and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, and the Windows™ logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.