eRoom Connector

Software Version 12.0.0

Release Notes

Document Release Date: June 2018

Software Release Date: June 2018

Legal notices

Copyright notice

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Trademark notices

Adobe™ is a trademark of Adobe Systems Incorporated.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

UNIX® is a registered trademark of The Open Group.

Documentation updates

The title page of this document contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To verify you are using the most recent edition of a document, go to https://softwaresupport.softwaregrp.com/group/softwaresupport/search-result?doctype=online help.

You will also receive new or updated editions of documentation if you subscribe to the appropriate product support service. Contact your Micro Focus sales representative for details.

To check for new versions of software, go to https://www.hpe.com/software/entitlements. To check for recent software patches, go to https://softwaresupport.softwaregrp.com/patches.

The sites listed in this section require you to sign in with a Software Passport. You can register for a Passport through a link on the site.

Support

Visit the Micro Focus Software Support Online website at https://softwaresupport.softwaregrp.com.

This website provides contact information and details about the products, services, and support that Micro Focus offers.

Micro Focus online support provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the support website to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Access the Software Licenses and Downloads portal
- Download software patches
- Access product documentation
- Manage support contracts
- Look up Micro Focus support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require you to register as a Passport user and sign in. Many also require a support contract.

You can register for a Software Passport through a link on the Software Support Online site.

To find more information about access levels, go to https://softwaresupport.softwaregrp.com/web/softwaresupport/access-levels.

Contents

New in this Release 5

Resolved Issues 10

Supported Operating System Platforms 11

Notes 12

Documentation 13

New in this Release

This section lists the enhancements to eRoom Connector version 12.0.0.

- The connector supports field standardization.
- The connector supports performance monitoring.
- The `QueueInfo` action has been improved:
  - The `QueueInfo` action supports additional parameters to filter and sort the actions that are returned when `QueueAction=getStatus`. For example, you can retrieve information about actions that have finished, and sort the actions by the time the connector finished processing them.
  - The `QueueInfo` action accepts a `MaxResults` parameter to restrict the number of results returned when `QueueAction=getStatus`. To page through the results, use the new parameter `Start`. For example:

    ```
    action=QueueInfo&QueueAction=getStatus&MaxResults=10&Start=1&QueueName=...
    ```

  - The `QueueInfo` action can return the action parameters that were passed to a pending or running action. You cannot retrieve the parameters passed to an action that has finished. To retrieve action parameters, use `QueueAction=getActionParameters`.
  - The connector returns the priority of queued actions when you use `action=QueueInfo&QueueAction=getStatus&QueueName=...`
  - The connector returns the status of a queue (paused or running) when you use `action=QueueInfo&QueueAction=QueueStatus&QueueName=...`
- The connector supports the `backupServer` and `restoreServer` actions. The `backupServer` action creates a backup of the connector's state. The `restoreServer` action restores the connector's state from a backup.
- The connector supports the `PurgeDatastore` action, which deletes the datastore for a fetch task. You might want to delete the datastore if you have modified your connector's configuration and you need to synchronize all of the documents again, even though the documents in the repository have not changed.
- The connector supports the `TaskList` action. This action returns information about the fetch tasks that are configured in the connector's configuration file, or that ran when the connector received a `fetch` action with a configuration included in the `config` action parameter.
- The connector can send documents to Haven OnDemand.
- The connector supports the `IngestBatchActions` configuration parameter. This parameter specifies actions to run on batches of documents after they are successfully sent to the ingestion server.
- The connector can index information directly into IDOL Server, a Vertica Database, or a MetaStore.

- Asynchronous action queues can be stored in an external database hosted on a database server, so that several installations of the component can share action queues.
- The `DeleteQueuedRequestsWhenServerStarts` and `DeleteProcessingRequestsWhenServerStarts` configuration parameters have been added. These parameters specify whether to remove queued and processing requests from asynchronous action queues when the server starts.
- The server requires less time to start when it is started for the first time and asynchronous action queues are stored in internal datastores.
- The connector includes Lua 5.3.0.
- The connector supports the `LuaDebug` action, which provides features for debugging your Lua scripts. The `LuaDebug` action can display the content of a script, and set and remove breakpoints in a script. When a script is paused at a breakpoint, it can return the values of the local variables, return the current call stack, and either step over lines of code or resume running the script. The connector also includes an example XSL template that transforms the output of the `LuaDebug` action. The template is named `LuaDebug` and can be found in the `acitemplates` folder.
- Asynchronous action queues can be stored in memory. This can improve performance in some cases.
- The connector supports the configuration parameter `IngestSourceConnectorFields`. If you set this to `TRUE` the connector adds fields to each document that identify the connector and fetch action that retrieved the document.
- The connector supports the actions `PauseSchedules` and `ResumeSchedules`, which pause and resume the starting of scheduled tasks.
- The `QueueInfo` action (`&QueueAction=pause`) can pause and resume single actions instead of an entire action queue. You can use this feature to pause specific fetch tasks.
- The connector provides additional statistics about the work it has completed, for example the number and frequency of ingest-adds, ingest-updates, and ingest-deletes. You can view these statistics through the `GetStatistics` service action. The connector also includes an XSL template that you can use to transform the output of the `GetStatistics` action and visualize the statistics.
- The `LogTypeCSVs` configuration parameter supports additional options for customizing logging. You can now create a separate log file for a fetch task or fetch action.
- The connector generates events to alert you when an asynchronous action queue becomes full, becomes empty, and when the queue size passes certain thresholds. You can handle these events with any of the existing event handlers.
- The connector can assign a priority to documents retrieved by a fetch task so that when they are ingested they are processed before documents retrieved by other tasks or other connectors. To use this feature set the new configuration parameter `IngestPriority`.
- The connector can send documents to NiFi Ingest, a new IDOL ingestion framework that is based on Apache NiFi. You can use NiFi Ingest instead of an IDOL Connector Framework Server.
- The connector supports the `schedule` action, which schedules actions to run automatically on a defined schedule.

- The connector supports the following Lua functions:
  - New functions and classes for parsing and manipulating JSON. The new functions are `parse_json`, `parse_json_array`, and `parse_json_object`. The new classes are `LuaJsonArray`, `LuaJsonObject`, and `LuaJsonValue`.
  - Functions related to sending HTTP requests and processing responses. These functions are provided by the `LuaHttpRequest` and `LuaHttpResponse` classes.
  - `addsection`, `getSection`, and `getSectionCount`. You can call these methods on a LuaDocument object to manipulate document sections. The existing Lua methods `appendContent`, `getContent`, and `setContent` now take an optional `number` argument that specifies the document section to use.
  - `base64_decode`, which decodes a base64-encoded string.
  - `base64_encode`, which base64-encodes a string.
  - `delete_path`, which deletes an empty directory.
  - `deleteFieldByPath`, which you can call on a `LuaDocument` or `LuaField` object to delete fields or sub-fields that match a specified path.
  - `doc_tracking`, which raises a document tracking event for a document.
  - `extract_date`, which searches a string for a date and returns the date.
  - `get_log`. This function returns a LuaLog object that you can use to write messages to a log stream configured in the connector's configuration file. When you use a LuaLog object the log stream settings such as the log file name, log level, and maximum size of the log file are respected.
  - `get_log_service`, and the new class `LuaLogService`. You can use these when you want to write log messages to a custom log file (instead of the standard ACI server log files).
  - `get_task_config`, which returns the configuration of the fetch task that called the script.
  - `get_task_name`, which returns the name of the fetch task that called the script.
  - `getFieldsByRegex`, which you can call on a `LuaDocument` or `LuaField` object to get fields or sub-fields where the name or path of the field or sub-field matches a regular expression.
  - `getValueByPath` and `getValuesByPath`. You can call these methods on a LuaDocument object or LuaField object. They return the value, or values, of a field or sub-field with a specified path.
  - `insertJson`, which you can call on a `LuaDocument` or `LuaField` object to add metadata from a `LuaJsonArray`, `LuaJsonObject`, or a JSON string to the document.
  - `LuaConfig:new`, which is the constructor that creates a new `LuaConfig` object.
  - `LuaDocument:new`, which is the constructor that creates a new `LuaDocument` object.
  - `parse_document_csv`, `parse_document_idx`, and `parse_document_xml`. These functions parse CSV, IDX, or XML files into documents and call a function on each document. `parse_document_idx` and `parse_document_xml` can also parse a string or file that contains a single document and return a `LuaDocument` object.
  - `regex_replace_all`, which searches a string for matches to a regular expression and replaces

all of the matches.

- `removeSection`, which is available on `LuaDocument` objects and removes a specified document section.
- `script_path`, which returns the path and file name of the script that is running.
- `send_and_wait_for_async_aci_action`. This function sends an action to an ACI server, polls the server until the action is complete, and then returns the response. This function is useful for sending asynchronous actions because it returns the action response instead of a token.
- `to_idx`, `to_xml`, and `to_json`. You can call these methods on a LuaDocument object. They return a string containing the document in IDX, XML, or JSON format.
- `url_escape`, which percent-encodes a string.

- The Lua function `get_config` no longer requires a filename. If you do not specify the file to load, the function returns the configuration file with the same name as the ACI server executable file.
- The Windows service installation (`-install`) now accepts a `-depend` command line parameter, which allows you to specify a comma-separated list of services that the service depends on. Windows services starts these dependencies before your service. For more information about installing Windows services, refer to the *IDOL Getting Started Guide*.
- The `SystemLimits` action has been added. This action returns information about system limits, such as the maximum number of open file handles, and the maximum map count (on Linux), which can affect the server.
- The server now logs the `ActionID` that it generates for an action that was sent without the `ActionID` parameter.
- You can now merge an external configuration file into the eRoom Connector configuration file. You can merge in a whole external file, a section, or a single parameter.
- The `ShowPermissions` action now shows the rules that define whether a particular origin IP has a particular type of permission. This information is returned only if you send the `ShowPermissions` action from a client that belongs to the admin authorization role.
- The `SSLMethod` configuration parameter now supports `TLSv1.2`.
- The `SSLCipherSuite` configuration parameter has been added. You can use this parameter to set an explicit list of ciphers to allow, or to disallow specific ciphers.
- The `SSLMethod` configuration parameter option `SSLV23` has been renamed to `Negotiate`. This option means that the server uses the highest available protocol in its SSL/TLS connections. The `SSLV23` name is still available, but might be deprecated in future.
- Action authorization is more flexible. The server supports the `[AuthorizationRoles]` configuration section and related parameters. You can create roles that can perform specific actions. You can identify the clients for each role by specifying client IPs and hosts, SSL identities, and GSS principals.
- You can now set `SSLCertificate` to be a chain certificate in PEM format (consisting of the end-entity certificate, any intermediate certificates, and ending with the root CA certificate). This option allows a complete certificate to be returned to the connected peer.

- You can now set `SSLCheckCerticate` to **`False`** even when `SSLCACertificate` or `SSLCACertificatePath` are set. This allows the component to fill in any chain required for the `SSLCertificate` by using the certificates that you specify in `SSLCACertificate` and `SSLCACertificatePath`, without requiring a certificate from the connected peer.
- The `GSSAPILibrary` configuration parameter has been added to the `[Paths]` section. You can set this parameter to the path to the GSSAPI shared library or DLL that the application uses. Depending on your system configuration, eRoom Connector attempts to detect the appropriate library to use. However, if you use Kerberos or GSSAPI security in your setup, Micro Focus recommends that you set an explicit value for this parameter.
- All ACI server ports now support the `Expect: 100-continue` HTTP header. Previously, third-party client applications that used this header (for example, using the cURL utility with the -F option to POST form data) could experience increased latency when communicating with eRoom Connector.
- When using GSS security, you can now configure the service to allow clients to authenticate to any service principal in the service's keytab, rather than requiring a single principal. You use this option by setting the `GSSServiceName` configuration parameter to an asterisk (`*`).
- The server can provide action responses in several different JSON formats. The default JSON response format (`ResponseFormat=JSON`) has been updated to use one of the new formats. For more information, refer to the documentation for the `ResponseFormat` action parameter.
- The OpenSSL library has been updated to version 1.0.2n.

Resolved Issues

This section lists the resolved issues in eRoom Connector version 12.0.0.

- The connector could terminate unexpectedly if a LuaField was assigned to a global variable.
- Asynchronous tokens were not always unique after a large number of requests, which could prevent scheduled tasks from starting.
- The Lua method `renameField` deleted the field when the old and new names were the same and the `case` argument was `true`, or the old and new names were the same (except for case) and the `case` argument was `false`.
- The Lua functions `send_aci_action` and `send_aci_command` ignored SSL parameters.
- In some circumstances, log messages related to ingestion could be repeated excessively frequently.
- The connector would not stop when requested, if documents were being ingested.
- The service port configuration did not correctly convert host names to IPv6 addresses.
- The `ShowPermissions` action did not return details for `ProxyClients`, `ServiceStatusClients`, and `ServiceControlClients` if these values were not explicitly set in the configuration file.
- The connector would not retrieve a license from a License Server with SSL enabled.
- The `GetLicenseInfo` action did not return the correct value for the `<autn:expirydays>` tag.
- The `GetVersion` action could incorrectly report the operating system on Microsoft Windows 10 and Microsoft Windows Server 2016.
- License related messages in the event log would appear from a different source to other messages.
- The `LogSysLog` logging configuration option did not output event logs.
- If an ACI Server was configured to request client SSL certificates, running multiple requests from a client could sometimes fail with **session id context uninitialized** errors. For example, this could occur when loading IDOL Admin.

Supported Operating System Platforms

The following operating system platforms are supported by eRoom Connector 12.0.0.

- Windows x86 64
- Linux x86 64
- Solaris x86 64
- Solaris SPARC 64

The most fully tested versions of these platforms are:

Windows

- Windows Server 2016 x86 64
- Windows Server 2012 x86 64
- Windows 7 SP1 x86 64
- Windows Server 2008 R2 x86 64
- Windows Server 2008 SP2 x86 64

Linux

For Linux, the minimum recommended versions of particular distributions are:

- Red Hat Enterprise Linux (RHEL) 6
- CentOS 6
- SuSE Linux Enterprise Server (SLES) 10
- Ubuntu 14.04
- Debian 7

Solaris

- Solaris 10
- Solaris 11

Notes

- The following configuration parameters, for action authorization by client IP address, have been deprecated:
  - `[Server] AdminClients`
  - `[Server] QueryClients`
  - `[Service] ServiceControlClients`
  - `[Service] ServiceStatusClients`

  You can now use the `[AuthorizationRoles]` configuration section to set up authorization for your servers more flexibly. These configuration parameters are still available for existing implementations, but they might be incompatible with new functionality. The parameters might be deleted in future.

Documentation

The following documentation was updated for this release.

- *eRoom Connector Administration Guide*
- *eRoom Connector Reference*