MANTA Flow Server Administration

Encoding

All property files are encoded in ISO 8859-1 character encoding. Characters that cannot be directly represented in this encoding can be written using Unicode escapes as defined in section 3.3 of The Java™ Language Specification; only a single "u" character is allowed in an escape sequence.

Authorization

This article describes the overall concepts of authentication and authorization in MANTA Flow Server. There are two guides that explain how to configure the authentication. Choose the right guide according to the version of MANTA you are using.

- For R34 and older, see Legacy Authentication (R34 and Older).
- For R35 and newer, see MANTA Identity Management: Keycloak.

User Roles

The application uses several user roles to authorize specific operations within the metadata repository. The roles are:

- ROLE_USER—basic role needed for all secured pages, which by default is all pages
- ROLE_EXPORTER—exports data from the repository
- ROLE_MERGER—executes data-modifying operations such as merging objects, truncating databases, and propagating edges
- ROLE_VIEWER_DATAFLOW—executes operations for the visualization of data flow
- ROLE_VIEWER_CATALOG—explores and searches the metadata catalog
- ROLE_USAGE—exports and cleans server-usage metadata
- ROLE_REPOSITORY_READ—executes data-reading operations via repository API
- ROLE_REPOSITORY_WRITE—executes data-modifying operations via repository API
- ROLE_REPOSITORY_EVALUATE—executes special data-modifying operations based on DSL-script evaluation via repository API
- Additional roles can be created simply by starting to use them as described below; creating new roles is only useful for the section called "Access Rights for the Metadata Repository"

Disabling Security for Specific Parts

> ⚠ Please note that as of MANTA R35 this change will not be persisted when upgrading to newer versions of MANTA.
>
> This configuration is no longer available as of R38.

It is possible to disable the authorization and authentication for specific parts of the MANTA Flow Server. For example, it is possible to view data flows without logging in.

The security definition is in the file `<MANTA_SERVER_HOME>/server/webapps/manta-dataflow-server/WEB-INF/classes/securityContext.xml`. (The configuration file is located in the `classes` folder, not the `config` folder, prior to R35.) There you will find a tag `sec:http` that contains access to the rules. Two lines of the configuration have to be added to allow anonymous access to some parts of the application.

- The URL pattern which should be permitted
  - `<sec:intercept-url pattern="URL_PATTERN" access="permitAll"/>`
  - For the dataflow visualization, the `URL_PATTERN` would be `/viewer/**`
- Roles which are granted to the anonymous user
  - `<sec:anonymous granted-authority="ROLE1,ROLE2"/>`

Access Rights for the Metadata Repository

It is possible to define access rights for the metadata repository. This means that some parts of the metadata repository may only be visible to particular users.

Enabling and Disabling Access Rights

To enable this feature, set the `repository.permissions-enabled` property in the configuration *Configurations* > *Server* > *Common* > *Repository Configuration* in MANTA Admin UI to true.

To disable this feature, set that property to `false`. In this case, the entire repository will be accessible to all users.

Defining Access Rights

If the access rights feature is enabled, it is possible to configure metadata repository permissions for MANTA users. This is done on three levels.

Level	Location of definition	Description
Repository view configuration	MANTA	A repository view is any part of the repository. It is defined as a set of included and excluded repository subtrees.
Assigning repository views to user roles	MANTA	Each role has a set of repository views assigned to it that are accessible to users with this role. It is possible to assign no view (i.e., no part of the repository is accessible) as well as the entire repository. One view can be assigned to more than one role.
Assigning users to user roles	External systems (e.g., LDAP) or MANTA	Each user can be assigned one or more roles and vice versa. There is no need to define the roles explicitly in MANTA.

Repository View Configuration

Repository views can be configured in the configuration *Configurations* > *Server* > *Common* > *Repository Views* in MANTA Admin UI. The rows are records of the inclusion of repository objects in the view or the exclusion of them from the view. The record fields are:

Repository view	Name of the repository view the record applies to
Type	Type of record. The value must be either `INCLUDE` or `EXCLUDE`. If it is `INCLUDE`, the affected objects are included in the view. If it is `EXCLUDE`, the affected objects are excluded from the view. The exclusions precede the inclusions. If only `EXCLUDE` records are defined for the view, the rest of the repository is considered to be included in the view.
Affected objects	Case-insensitive regular expressions of repository path entries separated by slashes (`/`). A repository object is affected if and only if: 1. each entry of its path matches the corresponding regular expression, 2. or any of its ancestors fulfill point one. The object resource is the first object path entry. Special cases: › To enclose a path entry in double quotes ("), use the `\"` sequence. › To use double quotes as part of the path entry, use the `\"\"` sequence. › To use a backslash (\) as part of the path entry, use the `\\` sequence. Note that a backslash in an object's path needs to be escaped twice—once for the CSV file and another time for the regex.

Assigning Repository Views to User Roles

The assignment of repository views to user roles can be configured in the configuration *Configurations* > *Server* > *Common* > *Repository Views Permissions* in MANTA Admin UI. The rows are records of view-to-role assignments. The record fields are:

Role	The user role the record applies to; must be unique within the file
Repository views	Comma-separated list of views accessible to the role; if set to `*`, users with this role have access to the entire repository

Applying the Changes

To apply changes to the CSV configuration files above, it is necessary to restart the MANTA Server or enter an HTTP GET request using the following format.

```
http://<server_name>:<port>/manta-dataflow-server/api/refresh
```

where the `<server_name>` and `<port>` are provided by your application administrator.

If the `repository.permissions-enabled` property has been changed, a MANTA Server restart is always necessary.

Repository Object Permission Evaluation

A repository object is accessible to the user if and only if it is contained in at least one repository view assigned to at least one of the user roles.

Example Configuration

repositoryViews.csv

```
"Repository View";"Type";"Affected Objects"
Teradata;INCLUDE;Teradata
OracleDwhExclParty;INCLUDE;Oracle/ORCL/DWH
OracleDwhExclParty;EXCLUDE;Oracle/ORCL/DWH/PARTY.*
MSSQLInstance;EXCLUDE;MSSQL/Server\\\\Instance/.*
```

repositoryViewsPermission.csv

```
"Role";"Repository Views"
ROLE_SYSTEM;*
ROLE_USER;Teradata,OracleDwhExclParty
```

Users with the `ROLE_SYSTEM` role have access to the entire repository.

Users with the `ROLE_USER` role have access to all Teradata databases and `ORCL.DWH` Oracle schemas, excluding all objects (tables, views, etc.) having a name starting with `PARTY`.

Repository

Back Up and Restore

Physical Copy

To back up the metadata repository:

- Ensure that no application is using the MANTA Flow Server.
- Stop the MANTA Flow Server application by either running the `bin/shutdown.(bat|sh)` script or stopping its installed service.
- Locate the database directory.
  - `webapps/manta-dataflow-server/WEB-INF/data/neo4j` (Neo4j database as of R36)
  - or `webapps/manta-dataflow-server/WEB-INF/db` (Titan database up to and including R35)
- Copy the database directory to a backup location.
- Start the MANTA Flow Server again by either running `bin/startup.(bat|sh)` or starting its installed service.
- Ensure that the MANTA Flow Viewer application is running on its configured URL.

To restore a previous backup of a metadata repository:

- Ensure that no application is using the MANTA Flow Server.
- Stop the MANTA Flow Server application by either running the `bin/shutdown.(bat|sh)` script or stopping its installed service.
- Locate the database directory.
  - `webapps/manta-dataflow-server/WEB-INF/data/neo4j` (Neo4j database as of R36)
  - or `webapps/manta-dataflow-server/WEB-INF/db` (Titan database up to and including R35)
- Remove the database directory.
- Copy the database directory from your backup location back to the original directory.
- Start the MANTA Flow Server again by either running `bin/startup.(bat|sh)` or starting its installed service.
- Ensure that the MANTA Flow Viewer application is running on its configured URL.

Physically copying the repository usually goes fast, but it depends on the OS and the version of MANTA Flow.

Dump

The metadata repository can be exported and imported in binary format, which is independent of the OS and the version of MANTA Flow. This dump applies to the whole repository including all versions and all technical metadata.

Dumps are only accessible to users with the role `ROLE_MERGER`.

Warning: Importing a dump will erase the old data in the target repository!

Links for the dump:

- Export – `<manta-server-url>/dump/export`
- Import – `<manta-server-url>/dump/import`

A warning message appears when exporting a repository greater than 4 GB. The repository size threshold can be configured before the server starts using the `minSizeForExportWarning` property inside the `webapps/manta-dataflow-server/WEB-INF/classes/dump-context.xml` file.

Please note that as of MANTA R35 this change will not be persisted when upgrading to newer versions of MANTA.

REST API

It is also possible to export and import dumps through the REST API.

- Export—`<manta-server-url>/api/dump/export`
  - Method: `GET`
  - The dump is returned in response as `Content-Disposition: attachment`
- Import—`<manta-server-url>/api/dump/import`
  - Method: `POST`
  - `Content-Type: multipart/form-data`
  - The dump is sent in an attachment named "`file`"
  - Returns JSON with "isOK" flag and an error message upon failure

Maintenance Links

There are several links to maintain the repository.

- `<manta-server-url>/api/truncate`—permanently erases all data in the repository so that it is impossible to restore without a backup
- `<manta-server-url>/api/inspect`—collects information about the last revision made and displays the numbers for each type
- `<manta-server-url>/api/revision/rollback-current-revision`—rolls back the newest revision, which must be uncommitted
- `<manta-server-url>/api/revision/delete-last`—deletes the last committed revision (fails if there is an uncommitted revision)

Location of Persistent Files

The configuration `Configurations > Server > Common > Repository Configuration` in MANTA Admin UI contains the locations of all repository files and directories that should persist across MANTA Flow Server reboots.

Property name	Description	Examples
repository.storage.neo4j	The relative or absolute path to the directory with the Neo4j metadata database (as of R36)	WEB-INF/data/neo4j c:/manta/data/neo4j
repository.storage.dir	The relative or absolute path to the directory with the Titan metadata database (up to and including R35)	WEB-INF/db c:/manta/db
repository.sourcecode.dir	The relative or absolute path to the directory containing source codes	WEB-INF/sourcecode c:/manta/sourcecode
repository.usage-stats.db	The relative or absolute path to the database containing usage statistics; should not contain the database file extension	WEB-INF/usageStatsDb c:/manta/usageStatsDb
repository.script-metadata.db	The relative or absolute path to the database containing script metadata; should not contain the database file extension	WEB-INF/scriptMetadataDb c:/manta/scriptMetadataDb
repository.permissions.views	The relative or absolute path to the file containing the configuration of views of the metadata	WEB-INF/conf/repositoryViews.csv c:/manta/repositoryViews.csv
repository.permissions.view-permissions	The relative or absolute path to the file containing the permission configuration of views to the metadata	WEB-INF/conf /repositoryViewsPermission.csv c:/manta /repositoryViewsPermission.csv

repository.horizontal-filters	The relative or absolute path to the file containing the configuration of horizontal filters	WEB-INF/conf/horizontalFilters.json c:/manta/horizontalFilters.json
repository.horizontal-filter-groups	The relative or absolute path to the file containing the configuration of horizontal filter groups	WEB-INF/conf/horizontalFilterGroups.json c:/manta/horizontalFilterGroups.json

Advanced Configuration (Titan Database up to R35)

The advanced repository configuration is also available on the MANTA Admin UI page `Configurations > Server > Common > Repository Configuration`.

Property name	Description	Default value
storage.backend	Storage backend to be used for persistence (full class name of the StorageManager implementation or one of the pre-defined storage backends)	persistit
storage.buffercount	Size of the Persistit internal buffer	5000
storage.transactions	Enables transactional isolation in a storage backend	false
storage.maximum-pages	A property from the underlying Persistit database that determines how big the database can be—if the number of pages is greater, the graph database crashes and has to be cleaned so everything works again The property must be set before the MANTA repository is created; it does not have any effect afterwards	1000000000
cache.db-cache	Enables the database level cache	true
cache.db-cache-size	The size of the database level cache (percentage of the total heap space available to the Titan JVM when < 1.0, absolute when > 1.0)	0.3
cache.db-cache-time	Expiration time in milliseconds for elements held in the database level cache	10000
index.node-name	This property enables/disables fulltext index used for search features in MANTA Flow Visualization and is turned on by default. It only makes sense to turn it off when the MANTA instance is only used for integration with third-party tools and the visualization is not (ever) accessed by users. Then it may improve the performance of the MANTA scan. The property must be set before the MANTA repository is created. It does not have any effect afterward.	true

We strongly recommend that you do not modify anything without first discussing it with the MANTA Support Team.

Session Policy

It is also possible to adjust the behavior of MANTA Viewer sessions. The file that contains this configuration is `<MANTA_SERVER_HOME>/webapps/manta-dataflow-server/WEB-INF/web.xml`. The default configuration appears as follows.

```
<session-config>
    <cookie-config>
        <http-only>true</http-only>
    </cookie-config>
</session-config>
```

It is possible to change the default session timeout by adding the following entry in *session-config*. The number inside the tag is the timeout in minutes.

```
<session-timeout>30</session-timeout>
```

Cookies

By default, whenever a new session is created, the server generates a cookie as well as the JSESSIONID on the URL. When the server verifies that cookies are allowed, the JSESSIONID isn't necessary and is dropped. If the client comes back with no cookie, then the server needs to continue to use JSESSIONID rewriting in URL.

This behavior can be adjusted by adding the following entry in *session-config*. If the tracking mode is set to *COOKIE*, the JSESSIONID is not generated at all. (MANTA Server has to be restarted so that the configuration is applied.) Please note that if this configuration is active, MANTA Viewer will not work with disabled cookies.

```
<tracking-mode>COOKIE</tracking-mode>
```

Version Control System

The metadata repository supports the version control system. This system can theoretically hold an unlimited number of revisions, but each revision takes up space on the disk. The maximum number of revisions is maintained by the prune feature, which deletes older revisions that exceed the configured amount.

It is possible to turn off the version control system. One reason for this could be to only use the MANTA Flow Server for exporting to third-party applications. In this case, the client workflow must be changed.

Revision System

Two types of revisions exist.

1. Major revisions—work like revisions in older versions of MANTA. Each major revision is identified by an integer.
2. Minor revisions—enable the incremental update of repository metadata. Loaded metadata is merged with metadata from the previous revision. Each minor revision is identified by a six-digit float number.

Basic API

- POST `<manta-server-url>/api/revision/createMajor`—creates a new major revision
- POST `<manta-server-url>/api/revision/createMinor`—creates a new minor revision
- GET `<manta-server-url>/api/revision/commit/<revision>`—commits the revision `<revision>`
- POST `<manta-server-url>/api/revision/prune-preserve`—prunes old revisions; request parameters:
  - `revisionCount`—count of revisions to be preserved
  - `majorsOnly`—if `true` (by default), only preserved revisions are counted over major revisions; if `false`, all revisions are counted

Logging

The MANTA Dataflow Server uses the log4j2 framework for logging. The configuration file is placed in `<MANTA_SERVER_HOME>/webapps/manta-dataflow-server/WEB-INF/classes/log4j2.xml`. The official documentation for this framework can be found here.

Usage Statistics

The MANTA Flow Server collects usage statistics in the inner database. These statistics can be exported using the link `<manta-server-url>/usage` as an archived CSV file. The export form enables you to set the starting date from which the statistics are exported. The statistics can be cleaned via the URL `<manta-server-url>/usage/clean`.

The usage statistics CSV file has the following columns.

- *date* —The date and time of the action that is logged
- *user* —The user who performed/triggered the action
- *action* —The code of the action that was performed; the table below contains a complete list of action codes together with descriptions of the actions
- *params* —Additional parameters that are logged; the attributes are represented as a JSON object and are different for each action. The key in the JSON object is always the name of the parameter, and the value is either a value literal or a nested JSON object.

Action source module	Action code	Action description	Params

MANTA Flow	`server_start`	This action is logged after the start-up of the MANTA Flow Server application.	› *system* – *osName*—OS name – *javaVersion*—Java Virtual Machine implementation version – *javaVnName*—Java Virtual Machine implementation name – javaRuntimeVersion—Java Virtual Machine implementation version including build ID › *mainConfig*—*the* content of the `<MANTA_SERVER>/webapps/manta-dataflow-server/WEB-INF/classes/repository.properties` configuration file
MANTA Flow Viewer	`flow_catalog_show`	This action is logged when the user views the main screen of the MANTA Flow Viewer application.	N/A
MANTA Flow Viewer	`flow_vis_refView`	This action is logged when the user executes the analysis and visualization of the data lineage in the MANTA Flow Viewer application.	› *refViewTime*—the time needed to calculate the data flow in ms › *refViewSize*—number of nodes contained in the data flow › *refViewError*—description of errors that potentially occurred when calculating the data flow › *initFollowTime*—the time needed to calculate the initial dataflow diagram that is visualized › *initFollowSize*—number of nodes contained in the initial dataflow diagram that is visualized › *isFromPermaLink*—flag whether the origin of the request is a permalink › *scopeMessage*—*a* message describing how the data flow had to be de-scoped, if it is too large to be visualized › *form*—values of all fields of the form whose sending initiated the dataflow request › *startNodeMap*—starting nodes used for the dataflow analyses › *flowStateGuid*—the technical state of the returned dataflow view
MANTA Flow Viewer	`flow_vis_contractionEvent`	This action is logged when the result of a data lineage analysis is too big to be visualized in MANTA Flow Viewer and has to be contracted.	› *message*—a message describing how the data flow had to be contracted
MANTA Flow Viewer	`flow_vis_aggEdgesIssue`	This action is logged when the result of a data lineage analysis contains too many aggregate edges to be visualized in MANTA Flow Viewer and, as a result, the aggregate edges have to be removed.	N/A
MANTA Flow Viewer	`flow_vis_sourceCode`	This action is logged when the user views the content of any source script in the MANTA Flow Viewer application.	› *node*—the node for which the source code is shown › *resource*—resource of the node for which the source code is shown › *flowStateGuid*—the technical state of the returned dataflow view
MANTA Flow Viewer	`flow_vis_permaLink`	This action is logged when the user generates a permalink for a specific data lineage visualization in the MANTA Flow Viewer application.	› *flowStateGuid*—the technical state of the returned dataflow view
MANTA Flow Viewer	`flow_vis_exportCsv`	This action is logged when the user exports the result of a data lineage analysis to a CSV. This export can be triggered either from the main screen or from the data lineage visualization screen of the MANTA Flow Viewer application.	› *flowStateGuid*—the technical state of the returned dataflow view
MANTA Flow Viewer	`flow_vis_exportPng`	This action is logged when the user exports a specific data lineage diagram to a PNG file. This export can only be triggered from the data lineage visualization screen of the MANTA Flow Viewer application.	› *flowStateGuid*—the technical state of the returned dataflow view
MANTA Flow Viewer	`flow_vis_exportDirect`	This action is logged when the user executes the analysis and export of the data lineage directly from the main screen of the MANTA Flow Viewer application. (The lineage is not visualized in MANTA Flow Viewer in this case.)	The params for this action are the same as for `flow_vis_refView` action.
MANTA Flow Merger	`flow_revision_new`	This action is logged whenever a new (major or minor) revision is created in the MANTA Flow Server repository.	› *revision*—the revision number of the newly created revision

MANTA Flow Merger	`flow_revision_commit`	This action is logged whenever an opened (major or minor) revision is committed in the MANTA Flow Server repository.	› *revision*—the revision number of the committed revision
MANTA Flow Merger	`flow_revision_rollback`	This action is logged whenever an opened (major or minor) revision is rolled back in the MANTA Flow Server repository.	› *revision*—the revision number of the rolled back revision › *result*—result of the rollback operation – *vertexActionMap*—map with operations performed over nodes – *edgeActionMap*—map with operations performed over edges
MANTA Flow Merger	`flow_revision_pruneTo`	This action is logged whenever the content of the MANTA Flow Server repository is pruned from the oldest revision to a specific (newer) revision.	› *perservedRevisionCount*—number of revisions that were preserved by the prune operation › *oldestCommittedRevision*—the oldest revision that was preserved by the prune operation › *latestCommittedRevision*—the latest revision that was preserved by the prune operation › *result*—result of the prune operation – *vertexActionMap*—map with operations performed over nodes – *edgeActionMap*—map with operations performed over edges
MANTA Flow Merger	`flow_revision_pruneOldest`	This action is logged whenever the content of the MANTA Flow Server repository is pruned and only a specified number of latest revisions is preserved.	The params for this action are the same as for the `flow_revision_pruneOldest` action.
MANTA Flow Merger	`flow_repository_truncate`	This action is logged whenever all the content of the MANTA Flow Server repository is truncated.	N/A
MANTA Flow Merger	`flow_repository_content`	This action is logged when the user views MANTA Flow Server repository statistics.	› *layers*—number of layer types stored in the MANTA Flow Server repository › *resources*—number of resource types stored in the MANTA Flow Server repository › *nodes*—number of node types stored in the MANTA Flow Server repository › *edges*—number of edge types stored in the MANTA Flow Server repository › *attributes*—number of attributes stored in the MANTA Flow Server repository › *revision*—revision number for which the statistics are counted
MANTA Flow Merger	`flow_repository_backLinks`	This action is logged whenever the backlink phase of the repository postprocessing scenario is executed for a specified MANTA resource.	› *revisionInterval*—the interval of revisions in which the operation is executed › *backLinks*—number of created backlinks
MANTA Flow Merger	`flow_repository_unification`	This action is logged whenever the unification phase of the repository postprocessing scenario is executed.	› *resourceName*—the name of the resource for which the was operation executed › *sources*—predicates used for identification of sources that should be unified › *targets*—predicates used for identification of targets that should be unified › *caseInsensitive*—flag whether to unify nodes without regard to their case › *forceMergeSources*—flag whether the unification of sources should be done even if there are more targets or no targets at all › *forceMergeTargets*—flag whether targets should be unified even if there are more of them › *result* – *unifiedObjects*—number of unified objects – *vertexMap*—map of counts of unified nodes per node type – *edgeMap*—map of counts of unified edges per edge type › *time*—the time needed for the unification
MANTA Flow Merger	`flow_repository_edgePropagate`	This action is logged whenever the edge propagation phase of the repository postprocessing scenario is executed.	› *revision*—revision in which the edges are propagated
MANTA Flow Merger	`flow_repository_contraction`	This action is logged whenever the contraction phase of the repository postprocessing scenario is executed.	› *result* – *vertexMap*—map of counts of contracted nodes per node type – *edgeMap*—map of counts of contracted edges per edge type

MANTA Flow Merger	`flow_repository_interpolation`	This action is logged whenever the lineage interpolation phase of the repository postprocessing scenario is executed.	› *result* – *interpolatedEdgeMap*—map of counts of interpolated edges per edge type – *skippedEdgeMap*—map of counts of skipped edges per edge type – *addedTransformationVertexMap*—map of counts of new transformation nodes per node type – *skippedTransformationVertexMap*—map of counts of skipped transformation nodes per node type – *errorTransformationVertexMap*—map of counts of failed attempts for the creation of new transformation nodes per node type
MANTA Flow Exporter	`flow_repository_export`	This action is logged whenever an export of the whole MANTA Flow Server repository is executed.	› *revision*—the revision in which the export is performed › *vertices*—map of counts of exported nodes per node type › *edges*—map of counts of exported edges per edge type

Module Specific Configuration

Configurations for the modules are located in the directory `<MANTA_SERVER_HOME>/webapps/manta-dataflow-server/WEB-INF/classes` under the name `<module-name>-context.xml`.

Merger

› Maximum leaf edges during edge propagations
  – Bean ID `edgePropagationHelper`
  – Property name `maximumLeafEdges`