HP Universal CMDB

for the Windows and Red Hat Enterprise Linux operating systems

Software Version: 10.00

Upgrader Reference

Document Release Date: June 2012

Software Release Date: June 2012

Legal Notices

Introduction

This document describes each step in the UCMDB upgrade process.

For each step, the following is described:

- A description of the step.
- Whether the step is critical. A step is considered critical in the following cases:
  - Skipping it would prevent the UCMDB server from starting after upgrade.
  - Skipping it would induce critical configuration or data loss that cannot be restored after upgrade.
  - Skipping it would prevent a critical component from operating properly after the upgrade.
- Whether the step can be re-run. In case of failure during the upgrade, whether or not this step can be re-run over the same schemas.
- Implications of failure. If this upgrade step fails, what is the effect on the UCMDB? If the step can be re-run, what can be done to resolve the issues?
- Log files: Important messages from the log file that are typical to this upgrade step, and the meaning of each message. Unless otherwise specified, all messages appear in the following log file: **C:\hp\UCMDB\UCMDBServer\runtime-upgrade\log\upgrade.short.log**

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
SchemaAdditionsUpgrader	Adds the new required tables and columns to the UCMDB.	✓	✓	UCMDB server cannot function properly if this upgrader fails.	upgrade.detailed.log
Global Settings Resources Upgrader	Upgrades all global settings from the old settings table (CCM_SETTINGS) to the new table: (URM_RESOURCES).	✓	✓	**Upgrade:** UCMDB server will not function properly **Cleanup:** CCM settings table will not be dropped. **Note**: This step cannot be rerun if the cleanup has already taken place because the old table is dropped the first time round.	• upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log
Customer Settings Resources Upgrader	Upgrades all customer specific settings from the old settings table (CCM_SETTINGS) to the new table: (URM_RESOURCES).	✓	✓	**Upgrade:** UCMDB server will not function properly **Cleanup:** CCM settings table will not be dropped. **Note**: This step cannot be rerun if the cleanup has already taken place because the old table is dropped the first time round.	• upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log
Database Privileges Upgrader	Checks if the mandatory 'create type' grant exists.	✓	✓	Fails the upgrade process with the following error message: "**The following privilege is missing: 'GRANT CREATE TYPE TO <USER_NAME>"**	• upgrade.log • upgrade.detailed.log • error.log
Class Model Resources Upgrader	Migrates all the class model resources from the old format (held in CCM_* tables in version 9.x) into the URM (Unified Resource Manager).	✓	✗	**Upgrade:** Old class model is not imported into the URM, runs once per customer. **Cleanup:** Deletes the CCM tables that are used in version 9.x to store the class model, runs once per schema	• upgrade.log • error.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
Class Model Extensions Upgrader	Copies the meta-data of class model extensions from the table where it was stored in version 9.x (table name CLASS_EXTENSION) into the (Unified Resource Manager).	✓	×	**Upgrade:** Class model meta data will not be imported into the URM, runs once per customer. **Cleanup:** Drops the CLASS_EXTENSION table, runs once per schema.	• upgrade.log • error.log
Consolidate Mode Upgrader	Consolidated mode is not supported as of ver. 10 of UCMDB. This upgrader 'isolates' consolidated customers.	✓	✓	**Upgrade:** The customer that was in consolidated mode in 9.x will not be isolated. **Cleanup:** The data (tables, views) about the consolidated customer will remain in the database	upgrade.detailed.log
Check cp and sp version are correct for upgrade	Ensures that the Content Pack and the service pack of the version from which the upgrade is being run are correct for the upgrade. It is meant to prevent upgrades on systems of unsupported versions of content and service packs. The upgrader does not actually make any modifications, it only performs a check. If the criteria are not satisfied, the upgrade fails.	✓	✓	The system configuration of the upgraded UCMDB is not correct. In this case, the upgrade cannot continue.	upgrade.log
Offline Resources Retriever	Extracts queries, views, reports, enrichments, correlations, folders and bundles from the database and stores them to disk. The resources are stored in **C:\hp\UCMDB\UCMDBServer\runtime\<Customer ID>\<resource type>**	✓	×	**Upgrade:** The failed resources will not be included in the environment after the upgrade. **Cleanup:** The resources will remain as garbage in the Model.	• upgrade.short.log • upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log Will contain the number of resources per type that were stored on disk, and warning and error messages in case there is a problem with a resource.

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
Update internal class model from resources	Updates the internal class model from the xml resources	✓	✓	Varies, depending on what exact resource has failed	• cmdb.classmodel.log • error.log
Upgrade reconciliation resources	Upgrades CIT identification rules and reconciliation priority definitions of external data sources.	✓	✓	CI reconciliation will not work properly when data is populating UCMDB.	• upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log
History Upgrader	• Runs the following steps for every CI type that is audited in the history. ○ Creates all the necessary monthly history tables (HDM/HDML) depending on the dates of the events for this CI type in the old history and the *history.purging.months.to.save.back* setting. ○ Upgrades each event that should not have been purged from the old history to the new history tables. ○ Inserts into the history root table a record for every CI that had at least one history event upgraded to the new history. • Purges the data while executing based on the *history.purging.months.to.save.back* setting, meaning that history events that should be purged would not be copied to the new history tables. • The *history.purging.days.to.save.back* setting from 9.0x is converted to months (rounding up) and is saved in the *history.purging.months.to.save.back*	✓	✓	If a CI type's history is not upgraded: • Model update and history operations will fail since history tables will not be created for this CI type. • No history events from the old history will be upgraded to the new history for this CI type. After the upgrade is finished and the server is running, you can run the **alignHistoryForType** operation from the history JMX services to create the history tables for a specific CI type that the history upgrader failed to upgrade or for all the CI types. You can also run **initializeHistoryDBFromModel** using the JMX console to create baseline records for all classes, based on current data model state. This operation ensures that queries return correct previous values and data of removed CIs. It may take several minutes and creates baseline events even for classes that upgraded properly (does not affect the history data correctness).	upgrade.detailed.log Logs the following: 1. Upgrade progress 2. History table creations 3. For every CI type, the number of CIs that will be upgraded and the progress of their upgrade. (debug mode)

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
	setting.			In this case, the Model update and history operations will not fail, but the new history will not contain any history events for this CI type.	
View Archive Upgrader	Copies view archive tables (VIEW_ARCHIVE and ARCHIVE_CHUNK) from the old history schema to the UCMDB schema. Archives are copied in bulks as they contain their TQL result.	✓	✓	**Upgrade:** The archive will not be copied to the new schema and will not be available in the upgraded system. **Cleanup:** The archives will remain as garbage in the old schema.	• upgrade.short.log • upgrade.detailed.log Will contain information about the archive being copied and the copy progress per-table.
Removed CIs History Upgrader	For every CI type that is audited in the new history, all the CIs that were removed in the time frame for saving removed CIs will be upgraded to the removed CIs tables (HDMR\HDMRL) of this CI type. The time frame for saving removed CIs is based on the *history.purging.months.to.save.back* and *history.purging.extra.months.to.save.back .removed.data* settings (new setting in 10.0, default value: 1 month). The time frame is from the date of the upgrade minus the number of months the purging saves date to the same date minus the number of months to save removed CIs. For example, . if <*history.purging.months.to.save.back*> is 3 months and *history.purging.extra.months.to.save.back .removed.data* is 2 months and the upgrade runs at 15.7.2012, the time frame will be from 15.2.2012 to 15.4.2012.	✓	✓	**Upgrade:** If a CI type's removed data history was not upgraded, the removed data history table will not be created or the removed data events from the old history will be missing in table. If the removed CIs history table was not created for a CI type, purging will not save removed CIs . When the upgrade is finished and the server is running, you can run the **alignHistoryForType** operation from the history JMX services to create the history tables for a specific CI type that the history upgrader failed to upgrade or for all the CI types. In this case, the removed CIs tables for all the CI types that the operation ran for will not contain any events from the old history. **Cleanup:** Old history tables will not be deleted from the old history schema	upgrade.detailed.log Logs the following: 1. Upgrade progress 2. Removed CIs History table creation 3. Each class number of removed CIs found in the time frame
Authorization Upgrader	Upgrades 9.0x roles and users, as follows:	✓	✓	**Upgrade:** • If failed while upgrading the roles:	security.authorization.management.log: Logs every modification to the authorization model, such as roles

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
	1. Upgrades roles: ○ Each 9.x role is converted to a 10.0 role (there is specific conversion for each internal permission) ○ If 9.x role1 is associated with 9.x role2, the corresponding upgraded role1 in 10.0 will contain the permissions of both role1 and role2 (because in 10.0 the are no role associations). 2. Upgrades users: ○ Upgrades each user's properties ○ If the 9.x user is associated with the 'admin' role, or with a role that is associated with 'admin' – creates an association between the corresponding 10.0 user and the SuperAdmin role. ○ In all other cases ▪ Creates an association between the 10.0 user and the 10.0 upgraded roles, according to the role associations it had in 9.x ▪ If the 9.x user had permission assigned directly to the user, then a new role is created in 10.0 with these permissions, and an association is created between the 10.0 converted user and this new role. The new role is named ConvertedRole1/2/3.... This name may be used by several 10.0 users that had the same permissions in 9.x. ▪ If a 9.x user had at least one discovery-related permission (whether directly assigned or via a role), then this user is granted the Discovery and Integrations Admin role in 10.00.			some of the roles that existed in 9.x will not exist in 10.0; no user will exist in 10.0 (except for the out-of-the-box users) • If failed while upgrading the users: some of the users that existed in 9.x will not exist in 10.0; others may have only some of the permissions they had in 9.x **Cleanup:** • Old CI types may remain in the class model: user, integration_user, acl_role, acl_attachment, together with the 9.x CIs representing the users and roles. This does not have any effect on the 10.0 security management. • The obsolete Basic_Security.zip package may remain in the Package Manager.	creation (with the exact permissions), user creation, user role assignments, and so on. Users and Roles moves resources to the URM and writes to these log files: • upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
	3. Upgrades integration users: ○ Upgrades each integration user’s properties ○ Assigns SuperAdmin role to each integration user				
Authorization System Users Upgrader	Upgrades system users (sysadmin, UISysadmin, and possibly others that were created by the user). The following steps are performed: 1. Upgrades the properties of the system user. 2. Marks the 10.0 converted user as “server administrator” 3. Assigns the SuperAdmin/NonUISuperAdmin role to the system user. The NonUISuperAdmin role is a new role created in 10.0, that is used for system users that did not have permission to access the UI in 9.x (like sysadmin). **Note:** 9.x system users are converted to 10.0 “regular” users. There is no “system user” concept in 10.0.	✓	✓	**Upgrade:** System users will not be upgraded. sysadmin and UISysadmin will exist in 10.0, because they are taken from the Basic_Authorization.zip package. But if the user created other system users in 9.x, they may not exist in 10.0 **Cleanup:** The SYS_USERS table will remain in the database; old 9.x system users will remain. This will have no effect on 10.0 security management.	• Upgrade.log: Logs the upgrade steps • security.authorization.management .log: Logs every modification to the authorization model, such as role creation (with the exact permissions), user creation, user role assignments, and so on. • upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log
LDAP Group Mapping Upgrader	Upgrades LDAP mappings. In 9.x each LDAP user was mapped to a role. In 10.0, each user is mapped to a user group, and the user group is mapped to a role. The upgrader performs the following: 1. Collects all 9.x roles assigned to LDAP users. 2. For each 9.x role, creates a user group in 10.0, and assigns it to the corresponding 10.0 role. 3. Assigns each 10.0 user to a 10.0 user group.	✓	✓	LDAP users will not belong to user groups, and thus will have no permissions	• Upgrade.log: Logs the upgrade steps • security.authorization.management .log: Logs every modification to the authorization model, such as role creation (with the exact permissions), user creation, user role assignments, and so on.

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
LDAP Default Role Upgrader	Creates a new user group in UCMDB and assigns the previously upgraded role that was configured as the LDAP default role to this group.	×	✓	LDAP default role will be assigned when authenticating with an LDAP user that is part of an LDAP group that has no group mapping in the UCMDB.	upgrade.detailed.log
Offline Resource Loader	Loads the upgraded resources created in the "Offline Resource Retriever" step from the disk to the database.	✓	✓	**Upgrade:** The failed resources will not be included in the environment after the upgrade. However, factory resources will be included as they are deployed in a different step. Only the user resources are missing. **Cleanup:** The CI types that represent the old resources in the Model will remain in the Class Model.	• upgrade.short.log • upgrade.detailed.log Contains the number of resources per type that are stored on disk, and warning and error messages if there is a problem with a resource. **Note:** If you need to rerun this step you will get a warning for each resource that was already deployed the first time the step ran.
CIs Tenants Upgrader	Handles CI tenant assignments in a multi-tenant environment as follows: 1. Collects all tenant names from the **TenantOwner** and **TenantsUses** properties of all 9.x CIs. 2. Creates tenants with these names 3. For CIs that do not have these properties, sets the system default tenant as their tenant **"TenantOwner"** and "**TenantsUses"**	✓	✓	• Tenants that were defined on CIs in 9.x may not be created as "real tenants" in 10.0 • Some CIs may remain with no tenant owner – illegal state	• Upgrade log: Logs the upgrade steps • security.authorization.management .log: Logs tenant assignments to resources • upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log **Note:** CI tenant assignments are not logged
Resources Tenants Upgrader	In a multi-tenant environment, sets default tenant owner and consumers, as well as user owner, for all resources in 10.0. 1. For every 10.0 resource, sets its tenant owner and tenant consumer to be the system default tenant. 2. For every 10.0 resource, sets its user owner to be admin user.	✓	✓	Some resources may not have tenant owner or user owner – illegal state.	• Upgrade log: Logs the upgrade steps • security.authorization.management .log**:** Logs tenant assignments to resources

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
Tenants Information History Upgrader	For multi tenancy environments, for all the events that were upgraded, tenant owner/user will be entered: Tenant information (tenant owner and tenant user) was not tracked in version 9.x but is tracked in 10.0. Since there is no information from 9.x to enter in the tenant owner/tenant user columns in the 10.0 model, the upgrader does the following: • If there was any tenant information for the CIs that existed in the model, the tenant information is copied from the model. • For CIs that were removed and do not exist in the model, the tenant owner and tenant user columns will be populated with the default tenant in the system. **Note:** • Even if these attributes were set to being tracked by the history they will not be taken into account during the upgrade. • For single tenancy system the upgrade does not do anything.	✓	✓	Tenant information will be missing in the history events.	upgrade.detailed.log: • Logs upgrade progress • Logs all the CIs that were not found in the model and enters the default tenant for them (debug mode).
Federation URM Resources Upgrader	Upgrades the adapter, integration point and data push job resources to the URM and removes the Cmdb8xAdapter	✓	✗	All saved integration points and jobs will be lost and there will be no adapters available.	• upgrade.detailed.log • urm.log • urm_detailed.log • urm_audit.log • urm_detailed_audit.log
Basic Packages Deployer	Deploys packages related to the UCMDB	✓	✓	Varies, depending on what exact	• log\package_reports\customer_1\

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
	infrastructure.			resource has failed	deployReports • Mam.packaging.log • error.log • cmdb.classmodel.log
CP Deployer	Deploys the Content Pack provided with the new version	✓	✓	Varies very much on what exact resource has failed	• log\package_reports\customer_1\deployReports • Mam.packaging.log • error.log • cmdb.classmodel.log
CmdbAdapter Integration upgrader	Updates the CmdbAdapter integration and population jobs to suitable for the Cmdb9xAdapter.	✓	✗	The saved integrations and jobs that were defined for the CmdbAdapter will be lost.	upgrade.detailed.log
Consistency Upgrader	Fixes consistency issues in the database.	✓	✗	Database may contain inconsistent data (links with missing ends, IDs which exist only in the root table, or IDs which exist only in the CIT's data table)	cmdb.db.tool
OIDToHostClass upgrader	Upgrades oidToHostClass user-defined rules to the new rule format and saves the new rules in the **UserDefinedRules** package with the name, **userDefinedOidToHostClassRules**. Removes the old **oidToHostClass** resource from the Network package. Notes: • Part of the oidToHostClass upgrade is to recognize the user-defined rules and transform only them to the new discovery rules format (the out-of-the-box rules are not taken into consideration in this specific process). This recognition is done by comparing the oidToHostClass.xml content under CP.zip with the oidToHostClass resource deployed. The upgrader removes all of the out-of the box content	✓	✓	• **Upgrade:** User-defined oidToHostClass rules will not be changed to the new normalization rule format and will not be part of the discovery rule engine. • **Cleanup:** The old oidToHostClass resource will not be removed.	upgrade.detailed.log: Messages logged: • "Start upgrading oidToHostClass.xml" • "DONE upgrading oidToHostClass.xml" • Errors that will be logged during the upgrade: ○ "Previous CP.zip was not found under <defaultCpFile.getAbsolutePath>. using default file from CP10." ○ "oidToHostClass resource was not found in CP.zip. using default file from CP10." ○ "FAILED Upgrading oidToHostClass.xml"

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
	from the deployed data, and works only on what is left. The 9.x CP.zip file is not available when doing the upgrade - since the upgrade process is done by installing UCMDB 10.00 from scratch without configuring a new schema and upgrading only the schema, and the CP.zip used for UCMDB 10.00 does not contain the oidToHostClass.xml file. To solve this issue: As part of the upgrade, the oidToHostClass.xml is taken from CP10.zip. If the 9.x server that is being upgraded used a more newer Content Pack, you should put the Content Pack used under: <server content dir>\content_packs_9x. The upgrader first looks for the Content Pack in the folder specified above. If it is not there, it uses the saved file from Content Pack 10.00. • The upgrader will add to each output attribute in the user defined rules the prefix “discovered_” (for example, discovered_vendor, discovered_model) This does not happen in the out-of-the-box rules since the values there are already normalized. If the user wants to have the rules insert values to real attributes and not “discovered” attributes – he should open the userDefinedOidToHostClass file, check that output values (that he defined on the 9.x server) are normalized and change the rule to use real output attributes (for example, change “discovered_vendor” to be “vendor”)				

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
SysObjectIdUpgrader	Adds a "." (period) to the beginning of the **sys_object_id** attribute of the Node CIs. This is necessary for consistency with newly discovered **sys_object_id** attributes which start with a "." (period). **Note:** If the **sys_object_id** attribute is empty or already starts with a "." (period), it is not changed.	✗	✓	The value of the **sys_object_id** attribute will not be consistent with the new discovered Nodes.	upgrade.detailed.log • Reports upgrader start • Reports the number of CIs to upgrade • Reports upgrader finish • In case of failure, reports the cause of failure
High Availability Upgrader	Upgrades the high availability related data in UCMDB	✓	✓	The high availability related data tables will not be consistent with the new UCMDB codebase. The high availability feature will not work.	upgrade.detailed.log
HistoryAutoCompletionUpgrader	Completes history changes data in the history tables, for changes made during other upgrade steps.	✗	✓	May cause low performance the first time a user gets history data for specific CIs.	• upgrade.detailed.log: Upgrader progress data log • history.log: History operation log
UndeployOldAdaptersUpgrader	Undeploys the old RMI and Changes adapters and updates the data push jobs that used these adapters as a source to use a dummy source. In addition, data push jobs that used the old data push engine and were configured to have the RMI adapter as the source (jobs that were saved as RMI/topology) are marked to run data push jobs with a full layout TQL.	✓	✓	The old saved data push integrations that were configured will not be available.	upgrade.detailed.log
Reconciliation priorities cleanup upgrader (LOA table cleanup)	In UCMDB 9.x the reconciliation priorities of some of the deleted CIs were left in persistence (database table “LOA”). This upgrader removes obsolete reconciliation priorities.	✓	✓	Obsolete reconciliation priorities may cause failures when data is populating UCMDB.	upgrader.detailed.log
CleanupRemoveColumnsUpgrader	Removes the old and obsolete tables and columns from the CMDB.	✓	✗	Garbage will be left in the database and the system will not be cleaned properly.	upgrade.detailed.log
Search Configuration Upgrader	New search-engine configurations are saved in URM, while configurations for	✗	✓	Out-of-the-box search-configuration	upgrade.detailed.log

Upgrader	Description	Critical	Can be rerun	Implications of Failure	Log Files
	versions of UCMDB Browser prior to version 1.7 were defined through qualifiers in the class model. This upgrader upgrades the search configuration to support the new search engine. **Actions taken:** Any CI Type with qualifier CMS_BROWSER_SEARCH / MODELING_ENABLED is added to search-ranking-configuration with Ranking-Priority 2 . Any attribute with qualifier CMS_SEARCHABLE_ATTRIBUTE is added to search-indexer-configuration and to search-ranking-configuration with Ranking-Priority 2. **Notes:** Runs only if the **Enable search** option was selected during upgrade (**cmdb.search.enabled** is set to **TRUE**). Class model does not change following this upgrader, the qualifier used for versions of the UCMDB Browser prior to 1.7 remains.			is used.	
Search Indexer Upgrader	Indexes all relevant CI attributes to the new search engine, based on the upgraded configuration. Runs only if the **Enable search** option was selected during upgrade (**cmdb.search.enabled** is set to **TRUE**). Is performed in two phases: (1) writing DB tables content to CSV files in the <UCMDB Installation directory>\search\index_csvs\customer<customer-ID>\ folder.	×	✓	CIs will not be indexed to Solr (search will not work)	search.log Progress is reported in logs only during CSV files writing and not during indexing.

Upgrader	**Description**	**Critical**	**Can be rerun**	**Implications of Failure**	**Log Files**
	(2) indexing the CSV files content to the new search engine. May take time to complete: approximately 1 hour per 1million CIs.				
Disable Aging Upgrader	Sets the **model.aging.is.aging.enabled** settings to **false** for all the customers	✗	✗	The **model.aging.is.aging.enabled** setting will be **true** after the upgrade. This could cause the deletion of CIs if the server is not used after the upgrade.	
Run database statistics	Runs database statistics to optimize performance	✗	✓	Database statistics will not be run; might have performance implications	upgrade.detailed.log

Troubleshooting and Warning Logs for Resource Upgrade

The following issues can occur during resource upgrade. Be aware that none of these issues cause the upgrade to fail but might impact which resources are upgraded.

- If a TQL was created in a previous version with an invalid layout, it is inserted during upgrade and does not fail the upgrade. Rather a warning is added to the log. These TQLs could have been created by copy/pasting a valid TQL and their xml then manually edited. You may see the following error message in the error logs:
  Pattern: <pattern name> has invalid layout: <stack trace>.

- You cannot define a grouping on a contact node of a perspective. However, there was no validation to prevent such a grouping prior to version 10.00. A view created in this way does not work correctly and is removed during the upgrade process. You may see the following error message in the **cmdb.upgrade.short** and **cmdb.upgrade.detailed** logs:
  You cannot define grouping on a contact query node.
  Failed to add CMDB_VIEW <view name> You cannot define grouping on a contact query node.
  This would appear in cmdb.upgrade.short and cmdb.upgrade.detailed logs.

- You cannot define a TQL, view, impact analysis rule, or enrichment with an empty space in its suffix. However, there was no validation to prevent creating such TQLs prior to version 10.00. A TQL that contains spaces in its suffix is not upgraded, an error appears in the log, and you must define the TQL again using a valid name. You can perform this step prior to upgrade. If these names are not fixed prior to upgrade, you may see the following error message in the **cmdb.upgrade.short** and **cmdb.upgrade.detailed** logs:
  Resource name is not valid.
  Failed to add <resource type> <resource name> Resource name is not valid.

- In multi-tenancy implementations, you cannot define TQLs with an @ character because the @ character is used internally in MT environments. If you are upgrading a single tenant environment to a multi-tenant environment and you have TQLs with the @ character, these tqls are not upgraded and are lost and a warning is added to the log. You will have to define these TQLs again using a new name. You can perform this step prior to upgrade.
  If these names are not fixed prior to upgrade, you may see the following error message in the **cmdb.upgrade.short** and **cmdb.upgrade.detailed** logs:
  Resource <resource name> contains the @ character, which is forbidden for multi tenant environment. The resource will be removed from the upgraded system. Please create it again with different name.

- If you have any folders that contain both a resource and a subfolder with the same name, the resource does not upgrade and a warning is added to the log. It is recommended to rename the sub folder or save the resource with different name prior to performing the upgrade. You may see the following message in the error logs:
  WARN: Failed to add <resource type> <resource name> duplicate subfolder or resource [<resource name>] in [<folder name>]