The Mainframe Software Partner
For The Next 50 Years

DevEnterprise
Best Practices

Release 16.05

Please direct questions about DevEnterprise or comments on this document to:

Compuware Customer Support

http://go.compuware.com/

Doc. JUL2016
June 17, 2016

Contents

How to Start Collecting Data

Defining collections for the Metadata Analyzer requires some planning and strategizing prior to their creation and execution, especially for the various source code/copybook collections (COBOL, Assembler, PL/I). For example:

- Identify how many source collections will need to be created.
- Organize source collections by application or by source code libraries.
- If the source code, copybooks and/or JCL are stored in CA Endevor, determine whether your site utilizes shadow datasets. Then determine whether these datasets have date/time stamps on the members.

Load Data into DevEnterprise

Only excerpts of this process are included in this Guide, so it is important to refer to the DevEnterprise Online Help for a more detailed illustration.

Step 1. Set Up Initial Collections with 1-5 Members

Start small – build a collection that only goes after 1-5 members. This will ensure that everything is in place as far as security for HCI on the mainframe and/or potential JCL errors in the mainframe Tops. This step will need to be completed for each type of collection needed for your site (COBOL, Assembler, PL/I, JCL, CICS, DB2, IMS).

Step 2. Create Connections/Collections on the DevEnterprise Server

Compuware recommends creating a TSO APPLICATION ID (i.e. XPDEVUSR) with a password that does not expire for use in CONNECTION definitions. If a CONNECTION uses the administrator's TSO ID, and the password expires and/or is changed on the mainframe, the password in the CONNECTION will have to be updated as well; otherwise any and all collections using that CONNECTION will fail.

There are 3 different types of CONNECTIONS – HCI, TCP/IP and DB2.

a. The HCI Connection is used for the following collection types:

   - COBOL
   - ASSEMBLER
   - PL/I
   - JCL
   - CICS.

b. The TCP/IP Connection type is used for IMS and DDIO collections.

c. The DB2 Connection type is only used for DB2 collections.

Step 3. Run Collections

Step 4. Review Collection Logs For Missing Programs/Copybooks

Review the collection log for warnings about missing entities. Locate them and add the library name to the collection

Step 5. Update and Rerun Collections as Needed

Rerun the collection and make sure there are no more warnings about missing items. Repeat this process for each different type of collection, and in the case of source code, one for each application if they use different source/copybook libraries.

Step 6. Load Data into DevEnterprise with 6-100 Members

Once the initial collection phase has been successful, build a collection that goes after 6-100 members. This should ensure that all of the necessary datasets have been defined to the collection for source code, copybooks, JCL and PROCS.

Step 7. Modify Collections on the DevEnterprise Server

The collections that were created in the initial collection phase can be modified to increase the number of members to be collected. At this point any security and/or JCL issues with DevEnterprise and/or HCI should have been resolved. By increasing the number of members being collected for each collection type, you can now verify that the collections have been defined with all of the necessary datasets so that all of the necessary entities (source code, copybooks, JCL, PROCS, etc.) can be located.

Step 8. Run Collections

Step 9. Review Collection Logs for Missing Programs/Copybooks

Review the collection log for warnings about missing entities. Locate them and add the library name to the collection.

Step 10. Update and Rerun Collections as Needed

Rerun the collection and make sure there are no more warnings about missing items. Repeat this process as in the initial collection phase for each different type of collection, and in the case of source code, one for each application if they use different source/copybook libraries.

Step 11. Load Data into DevEnterprise – Full Collection

After successfully completing collections with 6-100 members, go back to the collection definitions and modify the member name area to collect all of the programs/JCL needed.

Step 12. Modify Collections on the DevEnterprise Server

The collections can now be modified to now include all of the members you wish to collect for each collection type. Follow the same steps to modify the collections as in step 7.

Step 13. Run Collections

Now that you have completed all of the steps for a full collection, go back and edit all of the collections you have created and modify them to collect all of the members you wish to add to your database. Depending upon the number of entities you are collecting, this phase can take several days to complete.

Step 14. Review Collection Logs for Missing Programs/Copybooks

Review the collection log for warnings about missing entities. If possible identify the location of the missing entities so that you can modify the collection(s) created in step 11.

Step 15. Update and Rerun Collections as Needed

Rerun the collection and make sure there are no more warnings about missing items. Repeat this process as in step 10 for each different type of collection, and in the case of source code, one for each application if they use different source/copybook libraries.

Tips About Collections

Accessing CA Endevor

If your site uses CA Endevor to store source code, copybooks and/or JCL, there are two known PTFs available from CA that should be applied to Endevor:

- PTF T980124
- Solution R012153

Collecting from DB2

If you plan to collect DB2 information using the Metadata Analyzer, it is **required** that the DB2 Connect fat client, version 7.1 or higher, be installed on the same server as the DevEnterprise server component is installed.

Note: The DB2 Connect thin client does not contain the drivers necessary for DevEnterprise to communicate with DB2 on the mainframe.

Collecting from CICS

To collect information about your CICS regions you will need to define the JCL used to start the CICS region. A dataset name and/or member should be specified (for example, SYS2.CICS.JCL(REGIONA)). To run CICS collections, the Metadata Analyzer collection must have read authority—through the UserID specified in the related connection—to access the dataset that contains the CICS Startup JCL, the dataset pointed to by the DFHCSD DD, and the load library that contains the DFHSIT module. You may need to work with your CICS Systems Programmer to create a copy of the fully expanded JCL and have that copy placed in a PDS that the DevEnterprise Administrator's USERID has READ access to. DevEnterprise will not make any changes to the JCL.

Programs using Pre-Compilers

Normal pre-compilers for CICS and DB2 are accounted for in the analysis. But other preprocessors such as IDMS and custom, site specific ones, are not. Without preprocessing the code will most likely not pass analysis. In these cases there are the following alternatives you can use.

The first would be to run the affected programs through the preprocessor and store the output in a designated file. Then run your collection from this file. This will allow full analysis, but you will lose the ability to detect the Copybooks used in the program since they will already be resolved.

The second option is to compile the programs and put them in the Compuware Source Listing Dataset or DDIO. You can then use the DDIO Collection for them, but these programs will not only lose Copybook traceability, the analysis will also be different than using regular source. See the "*Differences in analysis between actual Source and a Listing from the DDIO*" document for more information on what to expect.

A third and lesser used option is to use the Program Analyzer preprocessor exit. The Program Analyzer lets you call a preprocessor to perform pre-processing on your COBOL or PL/I program. If enabled, you can invoke and call the preprocessor immediately after each source record is read from the input program. The preprocessor can then change the record, delete the record, add records, or leave the record unchanged. All but the deleted records then go on to the parsing and analysis phase of the Program Analyzer. The source file on disk is not changed. This is an exit that you would have to code and maintain yourself based on a supplied sample. It handles one line at a time so complex substitutions are not possible.

Small Collections vs. Large Collections

There are merits to having small collections and also to having large collections.

Small collections are required if you need to set up custom concatenation for the Include libraries. If the concatenation varies, then you will need a separate collection for each variance.

Small collections also have the advantage of having that collection entity attached to all entities in the collection. This is useful if you want to search for all programs in that particular application for example.

On the other hand, many small collections could make it harder to maintain. If you moved a copy library you would then have to go into each collection and change it.

Larger collections can be easier to administer, but you lose the ability to uniquely identify by individual collections and this would not work if you used different concatenations.

The process to convert from small collections to larger collections is simple.

1. Creating the large collection that will encompass many smaller collections.
2. Run this collection. It will recollect, but should not re-download (unless there are changes). It will detect that these entities are currently in the database and it will add a relationship of the collection entity to them. Each of these entities will now be related to two different collections.
3. Delete the small collections. When you do this a message appears stating that deleting the collections could have the effect of deleting everything related to the collection. This is acceptable because it will delete the relationship of the collection to every entity in the collection. If an entity is only related to that collection, which should not be the case here, then it will in fact delete that entity. After you have deleted the small collections you will be left with the one larger collection that you can use to maintain the repository.

Note: If at some point you want to reverse the process, simply perform these steps in reverse to create many smaller collections, run them, and then delete the larger collection.

Collecting all Versions of a Program that Resides in Multiple Libraries

If a collection includes multiple concatenations of source libraries and a given program exists in more than one of them, the Metadata Analyzer will only collect the first occurrence of a given program. Additional versions of the same-named program will not be collected. This allows you to use staging libraries and keep the currently used version of the program in the repository. If you need each of the versions of the program to be collected, set up separate collections to collect from each of the concatenated libraries.

Forcing a Re-Collection

After the initial run of a collection, any subsequent runs of that collection will only recollect entities that have changed since the prior collection run. There may be times when you want to force a collection to re-collect all entities, regardless of whether or not they have changed since the prior collection run. Perhaps the easiest way would be to execute `runCollector -d MDSHost MDSPort CollectionName` from a command prompt. The `-d` parameter will force this recollection Alternatively, you can make a change to the collection which would cause it to totally recollect all the entities. Refer to the Help topic “Creating Collections”, subtopic “To edit a collection” for information on the types of changes which would trigger a recollection. Even adding a space after a field will be enough of a change to put this into effect.

Automating Collections

Schedule a collection to run by using a program (such as Microsoft Task Scheduler) that invokes the following:

```
RunCollector <Metadata Server host name> <Metadata Server port number>
<collection name>
```

To schedule multiple collections to run, create a .BAT file with the following format. The call before each collection causes a collection start after the previous collection completes.

```
call RunCollector.bat <Metadata Server host name> <Metadata Server port
number> <collection name>

call RunCollector.bat <Metadata Server host name> <Metadata Server port
number> <collection name>

call RunCollector.bat <Metadata Server host name> <Metadata Server port
number> <collection name>

call RunCollector.bat <Metadata Server host name> <Metadata Server port
number> <collection name>
```

Merge Rules

Why Merge Entities?

Program entities can be collected by more than one collector type. Each collector type collects metadata specific to that collector type. In order to provide complete and correct relationship, charting, and impact analysis information, you need to merge program entities collected from different collector types that are logically the same program. For example, source program entities (collected from COBOL, PL/I, Assembler, or DDIO) should be merged with load module program entities (collected from JCL, CICS, or IMS). DevEnterprise provides both an automated and a manual process to accomplish this task.

Using the Merge Rule Manager

A merge rule tells the Metadata Analyzer to merge separate entities with certain property combinations into one new entity. The Merge Rule Manager does the following:

- Suggests merge rules for you to consider so you don't need to actively look for matching program names that are missing a source library or load library.
- Searches the application's data in the database and finds them for you.
- Allows you to add individual merge rules that you find, and to edit or delete existing merge rules.

1. From the Tools menu, select Merge Rule Manager. The Merge Rule Manager dialog box appears and displays information about existing merge rules.

   Note: The Merge Rule Manager menu option is only available to Administrators.

2. To search the database for merge rule candidates, click Suggest. The New Merge Rule Suggestions dialog box appears. This will show potential merge rules. If you agree with the suggestions, you can select and create the rules.

For more information on Merge Rules see the “Merging entities” Help topic.

Repository Analysis Report

The Repository Analysis Report allows an administrator to get a quick assessment of the Metadata Repository to identify any problems, such as missing copybooks, libraries without timestamps, or the need to create additional merge rules.

To create a Repository Analysis Report:

1. Run the `runRepositoryAnalysis.bat` file, which is located in the BINW folder where DevEnterprise was installed (`C:\Program Files\Compuware\DevEnterprise` by default). The batch application analyzes the entire repository and generates the Repository Analysis Report.

   Note: Only an Admin user can create a Repository Analysis Report.

2. View the report, which is named `RepositoryAnalysis_yyyymmdd_hhmmss.txt` and is located in the Reports directory in the User Data location specified during the installation (which is available in the About dialog box).

Use the report to improve your repository:

It is important to pay attention to the recommendations in the report. The following are areas where action is needed:

1. Missing Includes/Copybooks mean that a complete program cannot be formed so it cannot be learned. It will also mean that each collection will attempt to look for the missing Include/Copybook.
2. Missing Timestamps mean that each time a collection with that entity is run we will recollect it just to make sure we have the current version. Without a timestamp the collection can't be sure whether or not it has changed since it was last collected. This could mean more frequent collections than necessary. If the Include/Copybook is in a dataset that does not allow timestamps refer to "Bypassing Members Without a Date/Time Stamp During Program Collection" on page 11 for information on how to use the X2JM9DSN exit.
3. Potential Merge Rule pairs should be reviewed to see if use of the Merge Rule Manager is needed to automatically create new Merge Rules.

Metadata Server Database

The Metadata Server database is where the Metadata Analyzer stores collected entities. The Metadata Server serves as the interface between the Metadata Analyzer component and the database, allowing users to access metadata information stored in the database and display it using the Metadata Analyzer.

Maintaining the database

Compuware recommends that you maintain your database as follows:

- Stop the Metadata Server before applying a fix pack or running database maintenance.
- Perform backups of the database on a regular basis.
- Reorganize data and index pages on a regular basis. If you notice degradation in performance when collecting, learning, or performing searches, you should consider reorganizing data and indexing pages more frequently.
- If the option is available, specify that the pages are reorganized with the original amount of free space.

Bypassing Members Without a Date/Time Stamp During Program Collection

During program collection, the Metadata Analyzer's source collectors automatically collect members that have a changed date/time stamp. In addition, members that are missing a date/time stamp are recollected by default since it is not clear whether they have changed. Some copybooks/includes from IBM-supplied datasets do not change and recollecting these members increases collection time.

The X2JM9DSN exit allows users to indicate the dataset that contains the copybooks/includes that should not be recollected despite the fact that they are missing a date/time stamp.

For information on installing and using X2JM9DSN, review the source in SAMPLIB.

Note: Be prudent in your choice of datasets you add to X2JM9DSN. If they change while they are listed in X2JM9DSN, the Metadata Analyzer will not recollect them.

Security Considerations for DevEnterprise TPs

Batch TPs

HCI submits JCL within the PARMLIB member to the JES2/JES3 internal reader when it finds '//' in the first two positions of the first card of this member (Job Card). It first overrides the job name by substituting it with a unique name built from the characters specified in the SRVRJOB= field of the HCICNFIG suffixed with the number. For the Batch Job to be submitted the following security needs to be in place:

- UserID of the workstation user must be a valid ID.
- User ID of the workstation user must be authorized to submit jobs.
- UserID of the workstation user must have authorization to execute batch TSO (IKJEFT01 program).

Top Secret Users - A new facility needs to be placed in the facility matrix table (see *DevEnterprise Mainframe Installation and Configuration Guide* for exact control options). Workstation users need to be authorized to access this new facility (update the facility matrix table for each workstation user).

Started Task TPs

When TP runs as a started task, it first runs under the authority of a default user id. This default user id:

- Needs to be a valid user ID defined to the security system.
- Needs to have read access to the HCI authorized library where HCISERVR module resides.

The first step of the TP runs the HCISERVR program, which switches the authority of the task from the default user id to the actual user id of the PC user who initiates the TP from the workstation. Default user id is no longer needed by the task after HCISERVR finishes, since TP will continue executing under authority of the user id who initiated the task from the workstation.

Here at Compuware, we create default user id of XPEDJM50 and XPEDFT50, The names for these user ids happen to correspond to the names of started tasks that HCI will start when TPs are issues from the workstation. Started tasks XPEDJM50 and XPEDFT50 are made part of group STCGRP and user ids XPEDJM50 and XPEDFT50 are defined as aliases to the system catalog. This makes task XPEDJM50 to run under authority of user id XPEDJM50 and task XPEDFT50 to run under authority of XPEDFT50. If these users are not defined as aliases to the catalog, then the task runs under ++++++ user id.

The actual user id of the PC user:

- Needs authority to read all datasets used by the TP
- Needs to be a valid TSO user (have authority to issue TSO commands)

Security Considerations for HCI

For the HCI to function properly, it must be permitted READ access to the following:

- HCI authorized library
- HCI PARMLIB dataset
- HCI load library containing HCI configuration module

The HCI must be able to issue START, STOP, CANCEL, MODIFY and ROUTE operator commands.

To use HCI with TCP/IP, Open Edition MVS segment access needs to be defined.