Repository Attribute Mapping

When MDCMS receives committed files, it must determine which MDCMS attribute to assign to each file. Attribute mapping is the mechanism that makes this determination explicit and reliable.

Attribute Mapping is also used to determine the relative path for code when using Git Commit templates to sync changes from IBM i back to the origin repository. This is necessary to ensure that changes are committed to the correct location in the repository, and that the correct file extension is assigned to each file.

It is strongly recommended to define attribute mapping for all Git repositories. It is the fastest-performing mechanism, avoids potential path and file name resolution exceptions, and provides a clear overview of which attributes will be used for each path and file type.

In MDOpen: **Git Repositories → [select repository] → Attribute Mapping**

How MDCMS Resolves Attributes

For IFS / Remote Objects

1. Match on a path/file name pattern in the Attribute Mapping table for the repository.
2. Fall back to the Default MD Attribute defined on the CI definition.

For Source for IBM i Objects

1. Match on a path/file name pattern in the Attribute Mapping table.
2. If found in the Attribute Mapping Table, check MDCMS Object History for the most recent installation of the file into the same application for the same object type. If found, override the attribute from the Attribute Mapping table with the attribute used for that installation.

When not found in the Attribute Mapping table: 3. Check MDCMS Object History for the most recent installation into the same application and level. 4. Check MDCMS Object History for the most recent installation into the same application, regardless of level. 5. Check MDCMS Object History for any installation of the file, regardless of application or level. 6. In the MDCMS Attributes table, attempt to match on a combination of Folder and Default Source Type, or on a combination of Source Library, Source File, and Default Source Type to determine object type. If found, check for a reusable command for the object name and type; otherwise use the first matching MDCMS attribute. 7. Fall back to the Default MD Attribute defined on the CI definition.

Defining Attribute Mapping for an Individual Attribute

Click the **Add** link in the header to add a mapping entry from scratch, or click the **Copy** icon on an existing row to copy an existing mapping entry and adjust as needed.

Mapping Parameters

Appl

The target MDCMS application this mapping record applies to. A separate mapping record can be defined for each application targeted by CI definitions. Use content-assist to select from existing applications.

Path

A specific path from the root of the repository, or a wildcard pattern for matching paths. The path value does not include the file name itself. Use content-assist to select from existing paths in the repository.

Defining the specific folder path from the root of the Git repository is strongly recommended for each attribute. This is necessary when using Git Commit templates to sync changes from IBM i back to the origin repository, and when relative paths for object requests depend on the mapped attribute path.

File Name

A specific file name or a wildcard pattern for matching file names within the given path. For example, `*.sql` matches all file names with the extension of sql. Use content-assist to select from existing file names in the repository for the given path.

MD Attribute

The MDCMS attribute to assign when a file matches the path and file name criteria. Use content-assist to select from existing attributes.

Use `*NONE` to explicitly exclude any matching files from CI processing entirely.

Designated Git Commit Template Path for Attribute

If the same attribute is used for multiple paths during CI processing, setting this flag to true will indicate that this is the path to use when committing from MDCMS back to the git repository.

Default Attribute for Path/File Name

When multiple attributes are mapped for the same path/file name value, MDCMS will use the attribute by default where this flag is set to true. During CI processing, MDCMS Installation History will be checked to see if a different attribute should be used instead for the specific file.

For example, in path /source/dds for file name pattern *.prtf, the default may be to use the attribute named PRTF. However, certain printer files may need to be compiled with wider margins, and thus use the

PRTFWIDE attribute instead. By setting PRTF as the default attribute for the path/file name pattern, PRTF will be used for all files in that path with a .prtf extension except for those that have previously been installed with the PRTFWIDE attribute, which will continue to use that attribute.

Bulk Mapping Generation using the Missing Paths Feature

The fastest way to initially populate mappings is to click the **Missing Paths** link in the header of the Repository Attribute Mapping view.

> **Note:** The Missing Paths list depends on current data in the MDXREF database. Ensure that `*GIT` and/or `*IFS` are registered for cross-referencing and are up to date before using this feature.

Appl/Level for Attribute Mapping Parameters

Appl

The target MDCMS application the mapping records will apply to. Use content-assist to select from existing applications.

Level using Library/Folder Names

If the folder structure of the repository is organized in a way that includes the folder/library names and/or source file names, enter the level number of the application where the attribute definitions match the names used in the repository. Use content-assist to select from existing levels.

Repo Folder Structure

Select the option that matches the folder structure of the repository. This will determine how MDCMS parses the path and file name to determine the attribute.

- **none**: the repository does not have a consistent folder structure that can be used to determine attribute mapping, so the full path and file name will be used as-is for matching in the Attribute Mapping table.
- **Source Libraries**: the repository is organized with a folder structure that includes the source library name as the first folder level after the relative path under the root. For example, /source/APP1/ would indicate that files in that folder are associated with source stored in the APP1 library.
- **Source Libraries/Source Files**: the repository is organized with a folder structure that includes both the library name and source file name as the first two folder levels after the relative path under the root. For example, /source/APP1/QRPGLESRC would indicate that files in that folder are associated with source stored in the APP1 within the QRPGLESRC source file, and both values would be used for attribute mapping purposes.

- **Source Files**: the repository is organized with a folder structure that includes the source file name as the first folder level after the relative path under the root. For example, /source/QRPGLESRC would indicate that files in that folder are associated with source stored in the QRPGLESRC source file.
- **IFS Source Folders**: the repository is organized with a folder structure that includes the IFS source folder name as the first folder level after the relative path under the root. For example, /source/application-1/ would indicate that files in that folder are associated with source stored in the application-1 IFS source folder.
- **IFS Object Folders**: the repository is organized with a folder structure that includes the IFS object folder name as the first folder level after the relative path under the root. For example, /dist/application-1/ would indicate that files in that folder are associated with *IFS or *REMOTE objects stored in the application-1 object folder.

Repo Relative Path to Library/Folder Names

The relative path from the root of the repository to the folder level that contains the library and/or source file names used for attribute mapping. For example, if the repository uses a folder structure of /source/APP1/QRPGLESRC, and the relative path is defined as /source/, then MDCMS will parse the path to determine that APP1 is the library name and QRPGLESRC is the source file name for attribute mapping purposes.

Attribute Mapping - Unmapped Paths

After the **Next** button is clicked, the Unmapped Paths view will display each distinct path and file type in the repository that does not have a corresponding entry in the Attribute Mapping table based on the defined repository folder structure.

For each row where a matched attribute is found, the Select checkbox will be enabled.

If the attribute is not the correct attribute for the path/file type, or is set to "no match", the user can click the MD Attribute value to select a different attribute for the path/file type record.

Once one or more rows are selected, click the **Process Selections** link to process the selections. This will create new entries in the Attribute Mapping table for each selected row, using the matched or user-selected attribute. The new entries will be created with the path and file name pattern and will be assigned to the application defined in the previous step.

