# MDCMS Azure Pipeline and Release Interface

**MDCMS-Azure DevOps Interface Guide**

Published October 6, 2025

## Overview

There are 2 objectives of the Azure DevOps interface:

1) to invoke pipeline jobs defined on Azure DevOps servers to build, test, deploy, review and rollback artifacts on remote platforms from within an MDCMS RFP and to then capture the result and job details from those jobs
2) to approve Azure DevOps Releases to trigger the deployment of artifacts to target environments

This way, MDCMS acts as a centralized deployment manager across all platforms for an organization.

### Prerequisites

- MDCMS v8.4 or higher must be installed and licensed on at least one IBM i partition
- MDOpen v8.4 or higher must be licensed on the same IBM i partitions
- MDWorkflow Base + Pipeline must be licensed on the same IBM i partitions
- The user(s) responsible for configuring the interface must have access to MDOpen v8.4 or higher installed in MDOpen for RDi, MDOpen for VS Code or MDOpen for Web.
- The user(s) responsible for configuring the interface in MDOpen must be authorized to MDSEC Code 5 (Attribute Maintenance) and MDSEC Code 10 (Server Location Maintenance).
- The user(s) responsible for configuring the interface in Azure DevOps must have access to the Azure DevOps Dashboard and sufficient knowledge and privileges in Azure DevOps to:
  - create service users and add API tokens
  - add and update Azure DevOps Project Plans
- The network firewall must allow bi-directional http(s) traffic between the MDCMS partitions and Azure DevOps servers. The port to allow on the Azure DevOps side is the port that the Azure DevOps server listens to.
  The port to allow on the MDCMS side is either the port defined for the MD REST API server or the port defined for the http server that forwards requests to the MD REST API server.

 

- The credentials for an Azure DevOps user with admin rights to the applicable organizations/projects must be known.

## Configure Connection to an API Server

Before MDCMS can communicate with a Azure DevOps server, the location and credentials must be defined. The user that will configure this must have MDSEC authority to code md/10 – Server Location Maintenance.

Take the following steps to add an API Server connection:

1. Within MDOpen, connect to the repository for a partition and then expand Settings/DevOps Settings
2. Left-click on API Servers. The API Servers view will open and list any already defined servers
3. Within the view,select option Add (or Copy if a similar item already exists)
4. An editor will open with the following fields:

| | |
|---|---|
| Server ID | A 10-character field to uniquely identify the server definition.<br>The ID must be unique amongst all FTP servers and API servers.<br>The rename option can be used to change the ID on the definition and any setting that depends on the definition. |
| Description | A description of the server to make it easy to identify from a list |
| Server Type | The type of API server. Select Azure DevOps |
| Server URL | The URL of the server that MDCMS will use to communicate with using RESt APIs.<br>The URL should include the http://, the address and the port number if not the default http or https port.<br>For example: https://Azure DevOps.mycompany.com:1234 |
| User | A user id that is registered in Azure DevOps and has an API Token defined for it |
| Set New Token | Azure REST APIs are accessed using a personal access token. Take the following steps in Azure DevOps to generate a token:<br>1. Sign into Azure DevOps with the user<br>2. click on the User Settings icon at the top-right of the screen and select option Personal access tokens<br>3. click the + New Token link to create a new token |

| | |
|---|---|
| | 4. set the expiration as long as allowed. When a token expires, you can use the MDOpen option to update the token to easily apply a new token value to the existing server definitions.<br>5. Set the scope to either Full access or to Custom defined with Read, write & manage scope for Work Items. If also using the MDCMS Pipeline interface for Azure DevOps, also set the Release Scope to Read, write, execute & manage.<br>6. Click create and then copy/paste the token into the Set New Token field in MDOpen |
| Organization | The name of an Azure DevOps Organization – this name must exactly match |
| Project | The name of an Azure DevOps Project – this name must exactly match |
| Queue Nbr | this is only applicable when working with Work Items |
| Proxy Address | The address of a proxy server to route the HTTP connection through, if necessary |
| Proxy Port | The port number of the proxy server to route the HTTP connection through |
| Proxy Type | • HTTP<br>• SOCKS5 |
| Proxy User | The user id for the connection to the Proxy Server, if necessary |
| Set New Proxy Password | The password for the Proxy User, if necessary |
| Automatically Follow Redirects | For Azure DevOps in the cloud, it is important that this is set to true |
| Follow Authorization Header on Redirect | This should be set to false for Azure DevOps to avoid unauthorized access to credentials |
| Maximum Number | The value of 3 is recommended |

 

| | |
|---|---|
| of Redirects | |

Once the field values have been entered, click the Save button

### Test Connection to API Server

The connection can be tested by clicking on the Test Connection icon on a row in the API Server view.

The pass/fail message will be displayed in MDOpen and the detailed logging can be viewed by clicking on the Connection Logs icon.

The API consumer logs are stored as IFS files (one per day per server) in folder /MDCMS/logs/<instance>/pipeline. A log file is deleted after n days, based on the retention settings in the MDCMS Log maintenance screen. The default is to retain these IFS files for 10 days.

### Map Server Users to MDCMS Users

If setting up the triggering of Release Pipelines from MDCMS, then any MDCMS developers that are authorized to approve the deployment of a release must be mapped to Azure DevOps team members. To do this:

1. In MDOpen, right-click on the API Server and select option Project User Mapping
2. Click the Get Project Users option to retrieve the latest list of users belonging to the Project
3. For each user that will be used as a Release Pipeline Approver, and isn't yet mapped to the correct MDCMS user, click on the Server User Name and then enter the correct MDCMS user in the MDCMS user field or click the content-assist icon to select from a list.
4. Click Save

## Configure *PIPE Attributes

There is a special MDCMS Object Type, ***PIPE**, that can be assigned to an attribute. Attributes of this type indicate to MDCMS that 0 or more Pipeline jobs will be invoked for object requests assigned to the attribute when an RFP runs for a target level.

The same attribute ID should be defined for each level in a migration path so that the object requests will migrate from level to level. Then, for each level, Attribute Pipelines will be defined if any pipeline jobs should be invoked for that level. If no pipeline execution is required for a specific level, MDCMS will simply migrate the request records in the RFP without performing activity on them.

If certain artifacts will require a different set of Pipeline jobs then other artifacts, then a separate attribute should be created for them.

 

The user that will configure an attribute must have MDSEC authority to code 5 – Attribute Maintenance for the target Appl/Lvl.

Take the following steps to add a *PIPE Attribute:

- Within MDOpen, expand the repository for a partition and then expand Settings
- Left-click on Attributes. The Attributes view will open and list any already defined attributes based on the filter values.
- Within the view, right-click and select option Add (or Copy if a similar item already exists)
- An editor will open with many fields. Only the following fields are relevant for *PIPE attributes:

| | |
|---|---|
| Application | The Application ID |
| Level | The target promotion level in the application |
| Object Type | Set to *PIPE |
| MDCMS Attribute | A 10-character ID to identify the group of artifacts that will be deployed by pipeline jobs using this attribute |
| Description | An optional description of the attribute |
| Target Library | An optional value that can be used as a wildcard to pass to pipeline job parameters. This, along with other parameters, may make it easier to reuse the same pipeline jobs for multiple environments. |
| Require Approval | Set to true if an authorized user must approve an RFP after the Build/Test phase is complete before the artifacts can be deployed. If an RFP contains object requests for this attribute, it will wait for approval when it otherwise wouldn’t, if approval isn’t always required for the target level. |
| Acceptance Group Type | A user group type can be defined for the attribute and when defined, once objects of the given attribute are installed into the target level, a member of the group defined for the impacted project(s) must review and then accept the results of the installation before the RFP can continue to the next level in the migration path.<br>This group type will be in addition to any Acceptance Group Types defined for all objects for the level. This way, this additional acceptance is only necessary for these special attributes. For example, a pipeline job may need to perform automated testing on the deployed objects or a release manager may need to manually review |

 

| | |
|---|---|
| | the state of the artifacts on the remote platform in the test environment before the RFP can be promoted to Production. |

- Once the field values have been entered, click the OK button

## Configure Attribute Pipelines

Once a Pipeline Server and a *PIPE attribute is defined, Attribute Pipelines can be defined to specify which Pipeline Jobs should run for which attribute at each promotion level.

The user that will configure an Attribute Pipeline must have MDSEC authority to code 5 – Attribute Maintenance for the target Appl/Lvl.

Take the following steps to add an Attribute Pipeline:

- Within the MDOpen API servers view, select option Attribute Pipelines for the server in question.
- Within the view, select option Add (or Copy if a similar item already exists)
- An editor will open with the following fields:

| | |
|---|---|
| Appl | The Application ID |
| Level | The target promotion level in the application |
| MD Attribute | A *PIPE attribute to attach a pipeline job to |
| Phase | The phase of the RFP during which the pipeline job should be executed<br>• Build/Test – this phase coincides with the Submit or Bundle phase of the RFP, which is when objects are compiled and validated, but before any objects should be updated in the target environment.<br>• Deploy – this phase coincides with the Install phase of the RFP, which is when the target application is updated with new, changed or deleted objects. If the pipeline job is configured to perform the build and deploy at the same time, then set the Attribute Pipeline phase to Deploy. It is recommended to keep build separate from deploy, though, when possible, using conditional stages or multiple job definitions.<br>• Acceptance – this phase is executed when the RFP has completed installation in order to perform automated testing and acceptance of those tests against the target environment. |

 

| | |
|---|---|
| | • Rollback – this phase is executed automatically if the RFP fails at a step after the Deploy phase completed, or at a later date, if the successful RFP is rolled back and executed as a new rollback RFP. |
| Sequence | The sort sequence of the pipeline job when multiple pipelines are defined for the same attribute and phase. |
| Server ID | The ID of the defined Pipeline Server on which the Pipeline job will be executed. |
| Pipeline Job | For Build Pipelines, enter the Definition ID for the pipeline. To get the Definition ID:<br>1) Within the Project, select Pipelines<br>2) Click on the Pipeline that will be triggered by MDCMS<br>3) Click on the URL in the browser to view the full URL. A part of the URL will be definitionId=n. n is the value to use for the Pipeline Job.<br>For Release Pipelines, enter the name of the Pipeline. This name must exactly match the name in Azure DevOps and is case sensitive. |
| Pipeline Type | The type of Pipeline in Azure DevOps Build – The pipeline is a build pipeline, which is defined in Azure DevOps in the Project under Pipelines->Pipelines<br>Release – The pipeline is a release pipeline, which is defined in Azure DevOps in the Project under Pipelines->Releases |
| Stage | If for a Release Pipeline, the name of the Stage (Environment) that a release should be deployed to. This name must exactly match the name in Azure DevOps and is case sensitive.<br>The stage parameter is not applicable for build pipelines |
| Description | An optional description of the job |
| Run for Modifications | A checked value (Y) indicates this pipeline job should run for new or changed objects. |
| Run for Deletions | A checked value (Y) indicates this pipeline job should run for deleted objects. |
| Ignore Errors | A checked value (Y) indicating if the RFP should continue with warnings if the pipeline job execution fails. |

 

| | |
|---|---|
| Frequency | • Once per RFP – the pipeline job will run once for an RFP, if at least one request record of the given MDCMS attribute is assigned to the RFP.<br>• Once per Revision – the pipeline job will run once for each distinct set of object requests sharing the same Git Revision hash.<br>• Once per Folder – the pipeline job will run once for each distinct set of object requests residing in the same parent folder.<br>• Once per File – the pipeline job will run once for each object request of the given attribute. |
| Timeout in Seconds | The amount of time, in seconds, that MDCMS should wait for the pipeline job to respond as finished before MDCMS times out. A timeout will be treated as a warning or exception, depending on the Ignore Errors checkbox. |
| Run Concurrently | MDCMS can execute several different pipeline jobs concurrently, if those jobs aren't dependent on each other. A checked value (Y) indicates that this job can run concurrently with other jobs that are also marked as allowed to run concurrently. If this job, or an already running job doesn't allow concurrent execution, then this job won't be started until the currently running job has finished. |
| Workflow Acceptance Group | When phase Acceptance is selected, this required field is shown in order to register the defined User Group that contains the user to designate the accept/reject result to. The user group must be assigned to the project(s) impacted by the RFP's object requests as an acceptance role for either the target level or for the *PIPE attribute for the target level. |
| Workflow Acceptance User | When phase Acceptance is selected, this required field is shown in order to register the defined user to designate the accept/reject result to. The user must belong the Workflow Acceptance Group. The user must also be defined in MDSEC as either an actual user or as a System Process User. |

Once the field values have been entered, click the OK button

## Attribute Pipeline Parameters

Click the Pipeline Parameters icon on a row in the Attribute Pipelines view to manage parameters that are passed to the Pipeline job on the API server.

Attribute Pipeline Parameters are passed as Parameter/Value pairs to the job when it is requested to be executed.

 

These coincide with parameters that are defined for the Pipeline job that are then used within the Pipeline script instead of hardcoded values.

There is 1 String Parameter that is always passed by MDCMS and MUST NOT be additionally defined within MDOpen in the list of parameters for the Attribute Pipeline. That parameter is:

- MD_TRANS

Any other parameters that are required by your script should be defined in the pipeline YAML script and in MDOpen.

Attribute Pipeline Parameter fields:

| | |
|---|---|
| Parameter | The name of a non-MD Parameter that is defined for the Azure DevOps Pipeline job. The name must match exactly between MDCMS and Azure DevOps. |
| Description | An optional description of the parameter |
| Parameter Value | The value to pass to the job. This can be specific text, or a collection of text and placeholders.<br>You can prompt for placeholders by using F7. The selected placeholder will be inserted at the current cursor position.<br>***PASSTHRU** – this special value for a parameter indicates that the value should be retrieved from the result of a previously executed pipeline job for the RFP. |

Options to manage Parameters across Attribute Pipelines

Apply attribute pipeline parameter – when this option is selected, every Attribute Pipeline will be listed where the Parameter doesn't exist or the Parameter exists, but with a different description or value. Check the description or value checkbox for each Attribute Pipeline that you wish to apply the values of the parameter to.

Delete attribute pipeline parameter – when this option is selected (multi-select is enabled for this option), the current Attribute Pipeline will be shown at the top of the parameter deletion list with the Delete checkbox checked by default. Below that entry will be every other Attribute Pipeline where the parameter(s) are defined and the option to delete them from those entries as well.

## Define the MDCMS Service Connection in Azure

In Azure DevOps, MDCMS URL REST API is configured as a Service Connection which can then be used within a Pipeline Job. Steps to define the service connection:

1. Go to the Project Settings for the Azure DevOps project that contains the pipelines to be triggered by MDCMS.
2. Click on Service connections
3. Click the New service connection button at the top right
4. Select Generic as the connection type from the list and click Next
5. The Server URL will be the endpoint defined in MDCMS in the prior chapter + /mdcms + the callback resource depending on the Pipeline type.

   For Build pipelines, the resource is /pipeline/callback
   For example: https://mysystem.mycompany.com/mdcms/pipeline/callback

   For Release pipelines, the resource is /azure/release/callback
   For example: https://mysystem.mycompany.com/mdcms/azure/release/callback
6. The username and password can be left blank.
7. Enter a name for the Service connection. This name will then be used within the build or release pipeline
8. Set Grant access permission to all pipelines to true

## Add MDCMS Callback Stage to a Build Pipeline

Now that the Service Connection is defined, a final stage needs to be added to the YAML script for each Build pipeline in the project that is to be triggered by MDCMS.

1. In the Azure DevOps project, click Pipelines
2. Click on the pipeline to be triggered by MDCMS
3. Click the Edit button
4. Click the … to the right of the Run button and select triggers
5. Set Disable continuous integration to true, so that the pipeline isn't triggered automatically by a commit to a repo.
6. Return to the YAML editor
7. Append the following syntax to the end of the script to call the MDCMS REST API at the end of the job:

stage: md_update

displayName: Updating MDCMS

condition: always()

jobs:

- job: Update_MDCMS

 

```
pool: server

variables:

- name: MD_BUILDURL

value: $(System.CollectionUri)/$(System.TeamProject)/_build/results?buildId=$(Build.BuildId)&view=results

steps:

- task: InvokeRESTAPI@1

inputs:

connectionType: 'connectedServiceName'

serviceConnection: 'MDCMS_Callback'

method: 'POST'

body: |

{

"md_trans": "$(MD_TRANS)",

"summary_msg": "GET",

"build_url": "$(MD_BUILDURL)",

"build_num": "$(Build.BuildId)"

}

waitForCompletion: 'false'
```

Rename the highlighted Service Connection to the name you entered in the Project Settings.

If you wish to return passthru variable values to MDCMS for use in downstream Pipeline jobs, you should add the following syntax after the build_num variable in the body:

```
"passthru_name_1": "<your variable name>",

"passthru_value_1": “<your variable value>"
```

Up to 99 passthru name/value pairs are allowed to be returned to MDCMS.

## Add MDCMS Callback Job to a Release Pipeline

 

Now that the Service Connection is defined, a final job needs to be added to each stage for each Release pipeline in the project that is to be triggered by MDCMS.

1. In the Azure DevOps project, click Releases
2. Click on the Release to be triggered by MDCMS
3. Click the Edit button
4. Click the Pre-deployment Conditions icon for a stage
5. Enable Pre-deployment approvals and ensure that the Approvers list includes any MDCMS developers that would be authorized to request the release deployment to that environment. Each of those developers must be mapped for the Project.
6. The Approval order should be set to Any one user
7. Click on the jobs, tasks link for the stage
8. Click the … next to the parent process and select option Add an agentless job
9. Provide a job name such as MDCMS Callback
10. Set the Run this job parameter to Even if a previous job has failed
11. Click the + icon next to the job to add a task
12. Select Invoke REST API as the task type and click the Add button
13. Click on the new task
14. Provide a Display name such as MDCMS Callback
15. Set the Connection type to Generic
16. Select the MDCMS Service Connection for releases from the list
17. Set the Method to POST
18. Leave the Headers as they are
19. Paste the following JSON syntax into the Body parameter:
    ```
    {
    "release_url": "$(Release.ReleaseWebURL)",
    "release_id": "$(Release.ReleaseId)",
    "env_id": "$(Release.EnvironmentId)"
    }
    ```
20. Save the changes

## Requesting *PIPE Object Requests

Now that everything is configured, it’s time to start requesting items and having them built and deployed via Azure DevOps.

 

The most common way for files and folders to get requested is through Continuous Integration triggered by a push to a Git repository. This process is defined in MDOpen and full instructions on that configuration is available in the Continuous Integration Knowledge Guide.

If the Continuous Integration definition is set to create *PIPE request records, then those records will be created automatically for each new/changed/deleted file if for diffs, or for all folders and files if for contents.

Each record will contain the Repository type of *GIT, the Repository ID defined in MDOpen and the revision hash of the Git commit.

If manually creating *PIPE Object Requests, you can manually add the repository information to the Object Request Detail record in the Repository fields near the bottom of the Object Request editor.

If the pipeline job requires information about the SCM repository in order to pull artifacts from it, you can pass that information dynamically as wildcards on Attribute Pipeline Parameters.

SCM Wildcards:

| | | |
|---|---|---|
| ++GITBRN++ | Git Branch | The name of the Git Branch that an object request originated from. |
| ++GITREV++ | Git Revision | The Git Revision hash of the commit that an object request originated from |
| ++GITSVR++ | Git Server ID | The MDCMS Git server ID that an object request originated from |
| ++GITURL++ | Git Server URL | The Git URL defined for the MDCMS Git Server that an object request originated from |

MDCMS does NOT physically store and migrate actual files for *PIPE object requests, keeping the MDCMS process very light and fast. Instead, it utilizes best practices to expect the pipeline jobs to retrieve the files from the SCM themselves when they need it.

If using Continuous Integration or Cross-Platform impact analysis, MDCMS will clone the SCM repository onto the IBM i partition to perform tree-walking or diff-walking, but those files don’t proceed further in the promotion process.

Other Object Request-Specific Wildcards that may be useful as parameters:

| | | |
|---|---|---|
| ++OBJNAM++ | Object Name | The name of the requested object |

 

| | | |
|---|---|---|
| ++OBJDSC++ | Object Description | The object description value can be edited on the object request and modified at every level. |

## Requests for Release Pipelines

The triggering process of Release Pipelines by MDCMS is via the Approval of a Stage.

Once a release is generated out of a build pipeline, the exact name of the Release (case-sensitive) should be used as the name of the Object Request.

If the Release name is different for target environments higher up the migration path, the Object Request name can be renamed in MDCMS or MDOpen prior to submitting the RFP.

Pipeline Object Requests can also start higher up the migration path and don’t have to start at a checkout level.

 