Azure DevOps Work Item Interface for MDCMS

from Midrange Dynamics

Version 8.4

Published July 1, 2021

Azure Work Item Interface for MDCMS - Table of Contents

Overview

What is Azure DevOps?

Azure DevOps is a web-based, proprietary project and release management product, developed by Microsoft. It can be licensed to run in the cloud on Microsoft servers or licensed as a server product installed within the organization's infrastructure. The Azure DevOps interface for MDCMS is designed to communicate seamlessly with either licensing version with a common set of administrative tools.

MDCMS interfaces with Azure DevOps for the import of Work Item data into MDCMS Projects, Tasks and Subtasks. MDCMS also interfaces with the Git repositories in Azure DevOps and with the Pipelines. This document focuses on the interface for Work Items.

Information Shared between Azure DevOps and MDCMS

The intention of the interface is to enforce the workflow rules and information flow through Azure DevOps, so that Azure DevOps is the central Project/Task management tool for the organization and the full body of information about any given work item is easily reached from MDCMS, while the data stored within MDCMS is kept to the minimum necessary for developers to perform their work and for release managers to see the objects impacted by an Azure DevOps work item.

Projects outside of Azure DevOps can still be managed within MDCMS, if MDSEC authority is provided, if some aspects of development work are outside the scope of Azure DevOps.

A work item is imported into MDCMS as a Project, Task or Subtask with several key fields automatically mapped to MDCMS fields and any other fields can be optionally mapped to MDCMS custom fields.

Additionally, MDCMS can trigger state transitions for a work item in Azure DevOps and it can add comments to work items.

Prerequisites for using the Interface

- An active MDCMS license (v8.4+) on the IBM i partition used to connect to Azure DevOps
- An active MDOpen license (v8.4+) on the IBM i partition used to connect to Azure DevOps (a developer license for MDOpen isn't required for administrating the interface)
- An active MDWorkflow base license (v8.4+) on the IBM i partition used to connect to Azure DevOps
- An active MDWorkflow Pipeline license (v8.4+) on the IBM i partition used to connect to Azure DevOps
- The Azure DevOps URL must be reachable from the MDWorkflow partition via https
- The credentials for an Azure DevOps user(s) with admin rights to the applicable organizations/projects must be known.
- The administration of the Interface in MDOpen must be performed by a user with MDSEC authority to function code md/10 (Server Location Maintenance)

Optional Prerequisites to enable the MDCMS Webhooks for Azure DevOps:

- The MDCMS REST http server must be configured on the MDWorkflow partition (this is done from MDCMS->Setup Menu->Interface Settings->9= MDCMS REST API and Diagramming Server)
- The MDCMS REST http server on the MDWorkflow partition must be reachable from the Azure DevOps server

Configure the Interface in MDOpen

Preparation of MDOpen

1. Install MDOpen, which is the MDCMS plugin for Eclipse into an Eclipse IDE, such as Rational Developer for i or a flavor of eclipse from eclipse.org.
2. Create a repository connection in MDOpen to the IBM i partition that is designated as the MDWorkflow partition. This is typically the development partition, but may be a different one depending on firewall rules. That partition will then share the Work Item information with other partitions used by the same team, using the MDPUSH/MDPULL services.
3. Connect to the repository connection and expand the Settings section.

Pipeline Servers Settings

A Pipeline Server in MDCMS is the URL address and credential information used by MDCMS to communicate with a server over http(s) using REST APIs.

For this interface, there will be one Pipeline Server definition for each Azure DevOps Organization/Project combination that contains Work Items that should be imported into MDCMS.

Within the MDOpen Settings, click the option Pipeline Servers to manage the definitions.

Right-Click within the Pipeline Servers view and select option Add to add a new definition.

Left-Click on an existing definition to edit it. Right-Click on an existing definition for a number of other options.

Pipeline Server Parameters

Server ID	A unique ID of up to 10 characters to identify the server definition
Description	A description of the server
Pipeline Type	Set this parameter to Azure
Pipeline Server URL	The method, address and port number (if not default) of Azure DevOps server, not including any resource URI information in the path. If using the cloud version of Azure DevOps, this value will probably be **https://dev.azure.com**
User	The unique name (typically an email address) of a defined user in Azure DevOps with sufficient admin rights for the applicable Organization and Project. It is recommended, if possible, to have a special user defined such as mdcms@<myorg.com>, since any information pushed from MDCMS to Azure DevOps will be logged with that user.
Token	Azure REST APIs are accessed using a personal access token. Take the following steps in Azure DevOps to generate a token: 1. Sign into Azure DevOps with the user 2. click on the User Settings icon at the top-right of the screen and select option Personal access tokens 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. 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. 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	If you will be interfacing with several projects, it is recommended to use multi-threading in MDCMS to avoid latency. MDCMS can start up to 9 parallel MDAZUR jobs to communicate with Azure DevOps. For each Pipeline Server, select a queue number between 1 and 9 and have the projects spread evenly between the queues. Within Services->MDAZUR, set the # Parallel Jobs value to the highest queue number defined.
Proxy	If outbound traffic is routed through a proxy server, provide this information here
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 of Redirects	The value of 3 is recommended

Test/Troubleshoot Connection to Azure DevOps Server

Once a Pipeline Server definition has been saved, you can right-click on the entry and select option Test Connection. MDCMS will then start the MDAZUR service jobs (if not already started) and then test for the existence of the Organization and Project.

All REST communication from MDCMS to Azure DevOps is logged to the IFS with a log file per Server ID and day. Right-click on the Pipeline Server entry and select option Connection Logs to list and view the contents of the logs.

Project Mapping

For each Pipeline Server, which represents a Project in Azure DevOps, the various Work Item Types can be mapped to MDCMS Project and Task types. To manage this mapping, right-click on the Pipeline Server entry and select option Project Mapping.

This brings up a view with all defined Work Item Types for the project. If the list is empty, or if changes have been made to the types since the last pull of information into MDCMS, click the button Get Work Item Types. This may take several seconds. If it fails, review the Connection Log.

Left-Click on a work item type to edit the global mapping parameters for the work item type. Right-Click on a work item type to select the option to map individual fields and status codes.

Work Item Type Parameters

Enabled	When true, MDCMS will attempt to import work items of the given type. It is recommended to keep disabled until all of the fields, status codes and conditions are configured.
Import Level	Blank – work items of type should not be imported Project – the work items for this type should be imported into MDCMS as Projects Task - the work items for this type should be imported into MDCMS as Tasks Subtask - the work items for this type should be imported into MDCMS as Subtasks
Project Type	When Import Level=Project – the MDCMS Project Type to apply to imported projects for the work item type
Project Prefix	When Import Level=Project – a prefix string to place at the beginning of the Project ID in MDCMS. If used, this is followed by a – and then the work item number. If not used, the project ID is just work item number.
Project Digits	When Import Level=Project – the minimum number of digits to use for the Project ID. For example, if the work item number = 531 and project digits = 5, then the MDCMS project ID would be 00531.

Task Type	When Import Level=Task/Subtask – the MDCMS Task Type to apply to imported tasks for the work item type
Fixed MD Project	When Import Level=Task – the already defined MDCMS project to add the task to. If left blank, then the project will be the imported work item that is the parent in Azure DevOps. If the parent doesn't exist in MDCMS, then the task will be omitted.
Assignment Filter	This parameter filters the import of work items based on whom the work item is assigned to. optional – the work item doesn't have to be assigned to someone assigned to anyone – the work item has to be assigned to someone, but it doesn't matter who it is assigned to mapped MDSEC user – the work item has to be assigned to a user that is mapped to a user defined in MDSEC assigned to mapped MDSEC user with Role – the work item has to be assigned to a user that is mapped to a user defined in MDSEC that belongs to the role defined in parameter MDSEC User Role
MDSEC User Role	The role to check when Assignment Filter= assigned to a user that is mapped to a user defined in MDSEC
Wiql Query Conditions	Any additional conditions to apply for the import can be entered here in Wiql format. The conditions can be created and tested in Azure DevOps using the Wiql Playground. Then, copy those conditions into the MDCMS parameter. Only include text within the WHERE clause. Don't include the System.TeamProject or System.WorkItemType conditions, because MDCMS will automatically apply those.

Field Mapping for a Work Item Type

For each Work Item Type in a Project in Azure DevOps, the fields defined for the type can be mapped to fields in MDCMS. To view/manage the field mappings, right-click on the Work Item Type entry and select option Field Mapping.

Some of the fields have fixed mapping MDCMS fields.

The remaining fields can be mapped to either *intref for the Internal Reference field for a task/subtask, or they can be mapped to a custom field.

To modify the mapping for a row in the view, place the cursor in the MD Field Name field and edit the value. Content-Assist can be used to select a custom field ID from a list.

Work Item State -> MDCMS Status

For each Work Item Type in a Project in Azure DevOps, the work item import can be conditioned by the state of the work item and that state can be mapped to a MDCMS Project/Task status. To view/manage the State import rules and mappings, right-click on the Work Item Type entry and select option Work Item State -> MDCMS Status.

Editable Parameters

MD Status	The project or task status to map the Azure State to. If blank, then a work item in that state will not be imported as a new project or task in MDCMS. If the project or task already exists, then it will be updated, but the MD status will not be transitioned from its current value.
Update only	False – if the MD Status field is populated, a work item will be imported for a new or existing MDCMS project/task True – the work item will only be updated in MDCMS, but not newly created. The MD status will transition to the value in MD Status
No MD Transition if current Status	A set of current MDCMS status values can be selected. When the current MDCMS status for a project/task is equal to one of those values, then the project/task won't transition to the new status that is set in field MD Status. This is useful when a more granulated set of status codes are used in MDCMS vs. Azure DevOps.

MDCMS Status -> Work Item State

For each Work Item Type in a Project in Azure DevOps, the state of a work item can be automatically changed by MDCMS when the status of the corresponding project or task is changed.

To view/manage the State export rules and mappings, right-click on the Work Item Type entry and select option MDCMS Status -> Work Item State.

Editable Parameters

Azure State	If blank, then the manual transition of a project/task status in MDCMS to the given MD Status will not be allowed. If not blank, then an authorized user can change the status in MDCMS and the state of the corresponding Azure DevOps Work Item will be transitioned to the given value.

Internal only	If checked (true), then the manual transition of a project/task in MDCMS will be allowed, but the state of the Azure DevOps Work Item will not be modified.

User Mapping

For each Pipeline Server, which represents a Project in Azure DevOps, the team members can be mapped to MDSEC Users. To manage this mapping, right-click on the Pipeline Server entry and select option Project User Mapping.

This brings up a view with all defined users for the project. If the list is empty, or if changes have been made to the teams since the last pull of information into MDCMS, click the button Get Project Users. This may take several seconds. If it fails, review the Connection Log.

When the team members are retrieved from Azure DevOps, MDCMS will try to automatically map them to MDSEC users where the email address are the same or display names are the same.

To modify the mapping for a row in the view, place the cursor in the MDCMS User field and edit the value. Content-Assist can be used to select a MDCMS user from a list.

Project Webhooks

For each Pipeline Server, which represents a Project in Azure DevOps, webhooks can be created by MDCMS in the Azure DevOps Service Hooks settings in order for MDCMS to be automatically informed whenever a work item is created or updated, so that there isn’t a delay in importing the work item data into MDCMS.

See the pre-requisites section at the beginning of this tutorial to ensure that the communication from Azure to MDCMS will be feasible.

In order to create, update or delete the MDCMS webhooks for an Azure DevOps project, right-click on the Pipeline Server entry and select option Project Webhooks.

This brings up a view with both event types, and information about the currently registered Webhook, if applicable.

Press the Create/Update Webhooks to create the webhook for each event or to update them if the MDCMS URL changes due to a new address of the partition.

If the webhooks are no longer needed, then click the Delete Webhooks button.

In order to verify that the webhooks are functioning, do the following:

1. go into the Project in Azure DevOps
2. click on Project settings at the bottom-left
3. click on option Service hooks under the General section

4. verify that both events are listed and the host is correct.
   If they aren't listed, then create from MDOpen.
   If they are listed, ensure the State = Enable. If in a different state, then click on the … and Enable
5. click on History to be able to review any recent attempts made to invoke the webhook

Pull Work Items from Azure DevOps

Full Import

Once the mapping is complete for work item types, users, fields and status codes, enable the applicable Work Item Types in the Project Mapping view and then click button Import Work Items.

All work items of enabled work item types matching the filter criteria will be imported or updated in MDCMS.

A full import should be performed initially and whenever a work item type definition is added or modified.

Delta Import

Once an initial Full Import is performed, the MDAZUR service will check every n seconds for any new or changed work items since the last check for each enabled Azure DevOps Work Item Type that is registered in MDCMS. The bandwidth for this request is very small since only differences from the prior check are returned.

The number of seconds to delay is set in Services->MDAZUR

If using the MDCMS webhooks, the recommended delay interval is 1800 seconds (30 minutes)

If not using the MDCMS webhooks, the recommended delay interval is 30 seconds

Push Comments to Azure DevOps

Command MDCMS(ENV)/**MDAZURCMT** can be used to push information in the form of comments to Azure DevOps for either a specific Issue or for all impacted Azure DevOps Issues in an RFP. It is intended to let the Azure DevOps participants know about RFP activity in MDCMS and can be added as a *RFP command to occur at necessary exit points, such as after an install so that testing will proceed or when approval is required for an installation.

MDJIRACMT transactions are logged in file MDCMS(ENV)/MDDJIRR.

The MDCMS, MDXREF and MDSEC libraries must be in the library list when the command is invoked.

MDJIRCMT Parameters

Comment (CMT)	The comment to be added. The comment can be up to 1028 characters in

	length
Server ID (RSVR)	The Server ID of the server defined in MDOpen->Settings->Pipeline Servers. *RFP (default value) – the comment will be added for each Azure DevOps work item that is impacted by the RFP. Include the AGP and RFP parameters for this. If the command is used from an RFP exit point, set AGP to ##APPLIC## and RFP to ##RFPNBR##.
Workitem ID (WID)	A specific Workitem ID, when a specific Azure Server ID is Provided. Leave blank when RSVR = *RFP
Application (AGP)	The Application targeted by the RFP
RFP Number (RFP)	The RFP Number whose impacted Azure DevOps work item require a comment

Example of *RFP command to post a comment to impacted work items with a link to the RFP and its objects in the MDWorkflow WebApp:

Command Type = Q (Installation Archive/Cleanup)

Keep MD Libs in Libl = true

Command =

```
MDAZURCMT CMT('<p>Object changes have been deployed to the QA environment by MDC
MS RFP ##RFPNBR##</p><p><a href="https://<yourhost.com>/mdworkflow/pages/u
serLoginInit.jsf?link=true&screen=RW0011&location=##WFLLOC##&application=##APPLI
C##&RFP=##RFPNBR##">MDWorkflow Link</a></p>') AGP(##APPLIC##) RFP(##RFPNBR##)
```

View Azure DevOps Information in MDCMS

Azure DevOps Work Items as Projects

The Azure DevOps Work Item ID is used as the Project ID in MDCMS + any prefix set in the configuration, so searching and listing is the same as for any other project.

Azure DevOps Work Items as Tasks/Subtasks

The Azure DevOps Work Item ID is directly used as the Task or Subtask number. The only exception to this is if the Work Item ID is > 9,999,999 or if the task number already existed because it originated from somewhere else.

Additionally, the Work Item ID is stored in the MDCMS Task Reference Code. The code value can be searched and listed in MDCMS by pressing F8=Ref Code in the Task Listing.

In MDOpen, the Azure DevOps Workitem ID is shown in its own column in the task/subtask views and can be clicked on to open the page in Azure DevOps showing the Work Item.

