Cycle Bundle for Blue Yonder Warehouse Management

Upgrade Notes:
V3.1.0

Table of Contents

Introduction

This guide provides **valuable information about the installation and upgrade of the Cycle Bundle**. It also provides valuable information on the MOCA Environment variable changes needed as part of this release of the Cycle Bundle.

Cycle Bundle Installation and Upgrade Process

The following section will document the installation process for a **new install** of the Cycle Bundle as well as the process that **must** be followed if you are **upgrading** your existing Cycle Bundle to the **v3.1.0 release**.

New Installation of the Cycle Bundle

The Cycle Bundle is delivered as a ZIP file from the Cycle Portal. Extract the ZIP file into a newly created directory located on a local or network drive. It is then recommended that you import the contents of the Cycle Bundle into a **Source Control Management** (SCM) system and utilize that SCM repository for any further customizations or changes to your Cycle Bundle.

NOTE: All customizations should be stored in the Custom directories of the Cycle Bundle to maintain the integrity of the Cycle Base Bundle and to assist in a later upgrade process.

Upgrade of the Cycle Bundle from a version before v3.0

If you are upgrading from a version of the Cycle Bundle which is **earlier than v3.0.0**, please **consult your Tryon Solutions representative** for assistance with this upgrade.

Upgrade of the Cycle Bundle from a version v3.0.x or greater

If you have an existing version of the Cycle Bundle which is **greater than v3.0.0** and you are planning on upgrading to this **v3.1.0 release**, this process is **manual** and assumes that **all customizations/changes** you made to your existing Cycle Bundle have be stored in the **Custom directories of the Cycle Bundle**. If you made modifications to your existing Cycle Bundle and did not store the changes in the **Custom directories** (your directly modified files in the Base directories), **please stop and consult your Tryon Solutions representative for assistance**.

Assuming you made all your changes in the Custom directories of the Cycle Bundle and the **Base directories are unchanged** from the v3.0.x versions originally delivered, the process to update to v3.1.0 is noted here:

1. Ensure you have a **complete backup** of your **existing v3.0.x Cycle Bundle**.
   Both if on local/network drive or in a SCM system.
2. If you are using SCM, **we strongly recommend creating and checking out a new branch name (i.e. "v3.1.0")** and performing the activities below in that branch.

3. Install **v3.1.0** of the Cycle Bundle (from ZIP file) to a **new directory on a local or network drive**.

4. **Identify all Base directories** in the new **v3.1 Cycle Bundle**

   - An example Windows PowerShell command to find the **Base** directories in your **new v3.1.0 Cycle Bundle location**:

     PS> Get-ChildItem -Path **<PathTo3.1CycleBundle>** -Recurse -Include Base -Attributes Directory | Format-Table -AutoSize

   - **Base** directories exist in the following locations in the **v3.1 Cycle Bundle:**

     - BUNDLE_ROOT\Data\API\Base
     - BUNDLE_ROOT\Data\Dynamic Data\Base
     - BUNDLE_ROOT\Data\Interfaces\Base
     - BUNDLE_ROOT\Native App Locators\Base
     - BUNDLE_ROOT\Datasets\Base
     - BUNDLE_ROOT\Documentation\Test Specifications\Base
     - BUNDLE_ROOT\Playlists\Base
     - BUNDLE_ROOT\Groovy\Base
     - BUNDLE_ROOT\\MSQL_Files\Base
     - BUNDLE_ROOT\Test Case Validations\Base
     - BUNDLE_ROOT\Test Cases\Base
     - BUNDLE_ROOT\Utilities\Base

5. Create in your **existing v3.0.x Cycle Bundle location** the following directories. These are directories that are not in v3.0.x, but are in v3.1.0.

   - BUNDLE_ROOT\Data\API
   - BUNDLE_ROOT\Data\API\Base
   - BUNDLE_ROOT\Data\API\Custom
   - BUNDLE_ROOT\Utilities\Custom\Web
   - BUNDLE_ROOT\Utilities\Custom\Terminal
   - BUNDLE_ROOT\Utilities\Custom\API
   - BUNDLE_ROOT\Utilities\Custom\Mobile
   - BUNDLE_ROOT\Documentation\Test Specifications\Base (see note below)
   - BUNDLE_ROOT\Documentation\Test Specifications\Custom
   - BUNDLE_ROOT\Test Cases\Base\API
   - BUNDLE_ROOT\Test Case Inputs\Samples\v3.1.0

6. The v3.0.x Cycle Bundle directory **"Documentation/Test Specifications"** was now broken out into Base and Custom directories (this has been completed in v3.1.0). Within your v3.0.x directory, please use Windows File Explorer or similar tool to **move** your existing Specifications in "**Documentation/Test Specifications"** (directories (except custom) and files) **to "Documentation/Test Specifications/Custom".** This is to maintain those versions prior to v3.1.0.
7. For the Utilities/Base directory, please move/rename the **existing directory to Utilities/Base.SAVE**. This is need as the directory structure for Utilities/Base has changes from v3.0.x to v3.1.0. **Also, create a new Utilities/Base directory.**
8. After you have moved/renamed the Utilities/Base directory and have identified the Base directories and files to copy (and their associated files) from the v3.1.0 Cycle Bundle location, **recursively copy each Base directory and associated files (directory by directory)** using Windows File Explorer or another similar utility into your existing Cycle Bundle Location. Please note that you will be overriding files in your v3.0.x with files from v3.1.0, **you will need to answer yes to any override questions.**

   Note: The Utilities directory (Base and Custom) have been split out into separate directories for **Web, Terminal, API, and Mobile**. For any customizations (stored in the Utilities/Custom directory) that you have made **prior to v3.1.0**, after you have copied over the full Base directory into **Utilities/Base**, you will need to split/move your files out into the **separate Web and Terminal directories** in the **Utilities/Custom directory** (please key off "Terminal" or "Web" in the name of the Utility or look in the **v3.1.0 Cycle Bundle (or Utilities/Base)** for reference).
9. After you have installed the new Base versions into your existing Cycle Bundle location, recursively copy over the **"Test Case Inputs"** directory **from the new v3.1.0 Cycle Bundle location to Test Case Inputs/Samples/v3.1.0 in your existing Cycle Bundle area** using Windows File Explorer or another similar utility.
10. Next, we need to copy over data from the v3.1.0 Cycle Bundle location where we did not create a Base, Custom layer. These directories include:
    - Data/Samples
    - Data/Serial Numbers

    Copy the contents from the v3.1.0 Cycle Bundle location to same location (using Windows File Explorer or another similar utility) in your existing Cycle Bundle location.
11. The Environment_Template.csv files in the Environments directory **(Environments/Environment_Template.csv and Environments/WM201911/ Environment_Template.csv)** should be copied and upgraded from the v3.1.0 Cycle Bundle location to same location in your existing Cycle Bundle location (using Windows File Explorer or another similar utility).
12. Please see the section later in this document titled **MOCA Environment Variable Changes** and apply those changes to **your Custom Test Cases, Utilities, Datasets, and MSQL files**.
13. Please review **your Custom changes in all areas (especially Test Cases, Utilities, Datasets, Scripts)**, comparing your Custom versions to the new Base versions, **and integrating needed new functionality from the Base into your Custom code**.
14. Please review the list of changed Input CSV files (file with differences is in **"Test Case Inputs/Samples/v3.1/differences_versus_v3.0.x.txt"**) and **integrate the changes in these v3.1.0 changes into your existing Input CSV files** (to incorporate new changes to inputs).

15. After these steps have completed, please **execute a set of your Test Cases** to make sure they are functioning properly in your update Cycle Bundle area.

16. After everything has been updated and tests are working properly in the new v3.1.0 Cycle Bundle location, please take a **complete backup**

17. After everything has been updated and tests are working properly in the new v3.1.0 Cycle Bundle location, if you are using a SCM system, **please check all files into your SCM system** and back up your SCM data. If you used a **v3.1 branch, merge the contents of that branch in your main development branch**.

    You may consider adding a SCM label or other marker in your SCM system to denote the new v3.1.0 upgrade.

18. After these steps have completed, you can update your Environment files **for new features** such as API, Mobile, or Blue Yonder 2020 support.

MOCA Environment Variable Changes

As part of the support for Blue Yonder 2020 WMS, a **conversion of Test Cases, Utilities, Datasets, and MSQL Scripts is necessary** to support security changes in this release. These changes have been made in the **v3.1.0 version of the Bundle**, but the changes will need to be made to **your existing Custom Code** (code in Custom directories for Test Cases, Scripts/MSQL_Files, and Utilities)

The changes noted below are **backwards compatible to the Blue Yonder 2019 WMS release**, but please note that **Cycle 2.7.1 or greater is required** to implement these changes.

If you need assistance with these changes, **please consult your Tryon Solutions representative**.

Changes Needed

Some current datasets (MSQL load and cleanup code) do not use a publish data section at the top of the MSQL script. Moving forward all scripts interacting with input MOCA variables, **must have a publish data section at the top of the MSQL script** to declare the variables that will be used in the script.

For instance, an older Dataset (subset of) code could look similar to:

```
if (@@uc_cyc_lodnum = 'CYC-LOAD-XFER')
{
    generate next number
       where numcod = 'lodnum'
    |
      [select ftpcod
          from prtftp
          where prtnum = @@uc_cyc_prtnum
             and wh_id = @@uc_cyc_wh_id
             and prt_client_id = nvl(@@uc_cyc_prt_client_id,
@@uc_cyc_client_id)]
    |
    create inventory
     where srcloc = 'PERM-ADJ-LOC'
       and wh_id = nvl(@@uc_cyc_wh_id, @@wh_id)
       and reacod = 'ADJ-ACCEPT'
       and prtnum = @@uc_cyc_prtnum
       and prt_client_id = nvl(@@uc_cyc_prt_client_id, @@uc_cyc_client_id)
       and client_id = @@uc_cyc_client_id
       and lodnum = @nxtnum
       and dstloc = @@uc_cyc_srcloc
       and untqty = @@uc_cyc_untqty
```

You can see that MOCA environment variables set in your CycleScript code were used directly **with the "@@variable" syntax**. You can also see that no variables were declared in an initial publish data section.

An example of a conversion of this code would be:

```
publish data
  where lodnum = $lodnum
    and prtnum = $prtnum
    and wh_id = $wh_id
    and client_id = $client_id
    and prt_client_id = $prt_client_id
    and untqty = $untqty
    and srcloc = $srcloc
    and dstloc = $dstloc
    and usr_id = $username
|
if (@lodnum = 'CYC-LOAD-XFER')
{
    generate next number
       where numcod = 'lodnum'
    |
      [select ftpcod
           from prtftp
           where prtnum = @prtnum
```

```
          and wh_id = @wh_id
          and prt_client_id = @prt_client_id]
|
create inventory
 where srcloc = 'PERM-ADJ-LOC'
   and wh_id = @wh_id
   and reacod = 'ADJ-ACCEPT'
   and prtnum = @prtnum
   and prt_client_id = @prt_client_id
   and client_id = @client_id
   and ftpcod = @ftpcod
   and lodnum = @nxtnum
   and dstloc = @uc_cyc_srcloc
   and untqty = @untqty
```

Please notice that the (input) parameters used in the MSQL scripts are now in the same form as **CycleScript dollar variables (i.e. $wh_id))** and **not traditional "@@" MOCA environment variables (i.e. "@@wh_id") used prior**. Dollar variables will now be used for both variables used in CycleScript code and being used as input to the associated MSQL code used by that CycleScript code.

Because of this the **following CycleScript steps in Test Cases and Utilities, need to be removed from Test Cases and Test Utilities code** (v3.1.0 release of the Bundle has made these changes, but **you will need to make these changes in any Custom code that has been written**).

- I assign "<VALUE>" to MOCA environment variable "<VARIABLE_NAME>"
- I assign all chevron variables to MOCA environment variables **(currently called in every Test Case in Background scenario)**
- I assign all chevron variables to MOCA environment variables adding prefix "<PREFIX>"
- I assign dollar variables "<COMMA_SEPARATED_VARIABLE_NAMES>" to MOCA environment variables
- I assign dollar variables "<COMMA_SEPARATED_VARIABLE_NAMES>" to MOCA environment variables adding prefix "<PREFIX>"

An example of one of these changes would be that the following code would need to change:

```
Given I assign "prt-272" to variable "prtnum"
And I assign $prtnum to MOCA environment variable "uc_cyc_prtnum"

Then I assign "my_msql_script.msql" to variable "msql_file"
And I execute scenario "Perform MSQL Execution"
```

You would need to remove **"And I assign $prtnum to MOCA environment variable "uc_cyc_prtnum""** and your value of **$prtnum (set as a dollar variable) will be available to you in the MSQL script** as long as you use the following publish data block and follow-on statements:

```
publish data
  where prtnum = $prtnum
|
  [select stoloc from inventory_view
      where prtnum = @prtnum];
```

Notice that once variables are assigned in the publish data block, you can use as the MOCA variable as you always have (as “@prtnum”) in anything below the initial publish data block (meaning later in set of pipes or in a {} block).

Caveats and Exceptions

Caveat 1 - Sometimes because of the use of MSQL pipes and the MSQL statements executed (and the fact that calling MSQL commands in those pipe sequences can overwrite the data in the “@variable” variable), it is sometimes needed (by the MSQL script) to have the **original value from the publish data block** and **not from its current value relative to the MSQL pipe chain**. To achieve this, the current implementation of **Cycle (2.7.1+) stores the original value** (if declared in publish data block) **as uc_cyc_<variable>**.

For instance, if you have a block of MSQL code that has:

```
publish data
  where srcloc = $srcloc
    and dstloc = $dstloc
    ...
|
    create inventory
     where srcloc = 'PERM-ADJ-LOC'
       and wh_id = @wh_id
       and prtnum = @prtnum
       and prt_client_id = nvl(@prt_client_id, @client_id)
       and client_id = @client_id
       and invsts = 'A'
       and ftpcod = @ftpcod
       and lodnum = @nxtnum
       and dstloc = @uc_cyc_srcloc
       and untqty = @untqty
|
     create work
      where oprcod = 'TRN'
        and srcloc = @uc_cyc_srcloc
        and dstloc = @uc_cyc_dstloc
        and lodnum = @nxtnum
        and asg_usr_id = @usr_id
        and wrkpri = 1
        and lodlvl = 'L'
```

Because the MOCA command “create inventory” has **srcloc** as a parameter to its call and its value is set to **‘PERM-ADJ-LOC’** (which is not the original value passed in and published (in publish data block), the setting of “**and dstloc = @srcloc**” will use **‘PERM-ADJ-LOC’** at runtime and not it’s intended value which was passed in (**i.e. CYC_EA_LOC32**). To alleviate this issue, the **original value of srcloc** is used via “**and dstloc = @uc_cyc_srcloc”** and the original value **(i.e. CYC_EA_LOC32)** is used and not the modified value of **‘PERM-ADJ-LOC’**.

Caveat 2 - When using the **“@+” MSQL syntax,** you need to make sure you use **HIDE stack variable statement** to take into consideration case when the variable is never set.

For instance (original code):

```
publish data
    where wh_id = @@uc_cyc_wh_id
      and prtnum = @@uc_cyc_prtnum
      and stoloc = @@uc_cyc_stoloc
      and lodnum = @@uc_cyc_lodnum
      and prt_client_id = @@uc_cyc_prt_client_id
|
if (@@uc_cyc_subnum)
    publish data
       where subnum = @@uc_cyc_subnum
|
[select *
   from inventory_view
       where lodnum = @lodnum
         and stoloc = @stoloc
         and prtnum = @prtnum
         and @+subnum] catch(-1403)
```

and what new code would look like:

```
publish data
    where wh_id = $wh_id
      and prtnum = $prtnum
      and lodnum = $lodnum
      and stoloc = $stoloc
      and subnum = $subnum
|
if (@subnum = '')
   hide stack variable
       where name = 'subnum'
|
[select *
   from inventory_view
  where lodnum = @lodnum
    and stoloc = @stoloc
    and prtnum = @prtnum
    and @+subnum] catch(-1403)
```

In this example, **subnum is not set** in the context of current CycleScript code (in Input CSV or set in CycleScript code directly) but we are **now (in new code) are declaring it in the initial publish data section** (requirement of these changes). If the **hide stack variable** is not used (around subnum), then the resulting "select from inventory_view" will include "**and subnum is null"** from "@+subnum" and not the anticipated **"and 1 = 1"** substitution intended. This is because when declaring subnum in the initial publish data section, **the value is initialized** and is defined giving it a **different value/context in the "@+" statement**.

Caveat 3 - Because you no longer have (2) name spaces in the CycleScript code (one for MOCA Environment variables and another one for Dollar variables), you need to be careful on conversions that you don't blindly change **"I assign "<VALUE>" to MOCA environment variable "<VARIABLE_NAME>" to "I assign "<VALUE>" to variable "<VARIABLE_NAME>"** since in your original code you might have different values for these purposes and resetting the dollar variable value with what you used prior with MOCA may have unintended consequences.

For instance (original code):

```
Given I "Lookup reacod system code from description via MOCA"
    Then I assign $reacod to variable "code_description"
    And I assign "reacod" to variable "code_name"
    Then I execute scenario "Get Code Value from Code Description"
    And I assign $code_value to MOCA environment variable "uc_cyc_reacod"

    Then I assign "Web_Inv_Adjustment" to variable "dataset_directory"
    And I execute scenario "Perform MOCA Dataset"
```

and what new code would look like:

```
Given I "lookup reacod system code from description via MOCA"
    Then I assign $reacod to variable "code_description"
    And I assign $reacod to variable "save_reacod"
    And I assign "reacod" to variable "code_name"
    Then I execute scenario "Get Code Value from Code Description"
    And I assign $code_value to variable "reacod"

    Then I assign "Web_Inv_Adjustment" to variable "dataset_directory"
    And I execute scenario "Perform MOCA Dataset"

    And I assign $save_reacod to variable "reacod"
```

For More Information

Please see the Cycle User Guide and the MOCA section for more details on this subject.