## Existing Configuration Management

* CUSTOMER operates multiple QA environments.
* Each QA environment is supported by its own MC environment.
* Master config is used to deploy configurations to pre-preproduction and production environments.
* Configuration is typed into each QA specific MC environment independently.
* No automatic promotion of configuration from one QA environment to the next upstream environment such as pre-prod.
* A full deploy of configuration is made to the pre-prod/production environment when a QA stream goes live.



## Proposed Configuration Management

* Two QA environments to be provisioned in OMoC instance.
* Configuration for each QA environment will be maintained within the environment.
* At QA environment creation and at key project milestones (such as sprint completion), export configurations as XML files and store them in the CUSTOMER code repository.
* CUSTOMER discontinues the practice of full deployment of master configuration and convert to using configuration differences generated by the IBM CDT tool.

The above approach aims to solve the three key issues that CUSTOMER faces with configuration management:

* 1. Configuration differences to largely automate migration of differential configuration from a lower work stream to a higher work stream without re-typing.
	2. Configuration differences to automate removal of configuration from a higher work stream if it is promoted to production earlier than a lower work stream.
	3. Differences assist in application of hot fixes to the production and QA environments.

Below illustrates a OMoC environment with the following characteristics:

* One master configuration environment for Pre-Prod and Prod.
* Two QA environments. The QA environments do not have their own master configuration environments.
* At key milestones in the QA cycle, the configuration for the QA stream is exported as XML is stored in the code repository.
* Two development efforts are taking place concurrently at CUSTOMER:
	+ CUSTOMER Optimizer which began on January 01, 2021.
	+ Dev Ops which began on April 01, 2021.
* When CUSTOMER Optimizer workstream begin, QA1 was initialized with configuration from the Master Configuration environment.
* When Dev Ops workstream began, QA2 was initialized from the current state of the configuration in QA1. Hence, it contained production configuration plus all the work done on the CUSTOMER Optimizer stream between January 01, 2021 and April 01, 2021.
* At the time the CUSTOMER Optimizer project began, a full XML configuration export was stored in the code repository as “Q1-0.”
* At the time quality assurance began in QA2 on Dev Ops, full CDT XML exports were taken of the current state of CUSTOMER Optimizer (QA1) and Dev Ops (QA2).
* These full CDT exports were stored in the code repository as “Q1-1” and “Q2-0,” respectively.

**Use Case 1 – Managing Two Independent Work Streams**

After Dev Ops (QA2) is initialized with configuration from CUSTOMER Optimizer (QA1), sprint-2 for CUSTOMER Optimizer (QA1) starts and sprint-1 for Dev Ops (QA2) starts. In both QA1 and QA2, configuration is entered independently as required by the sprint.

CUSTOMER has a policy of merging code upwards whenever a sprint ends in a QA environment. In this case, CUSTOMER needs to merge the configuration from CUSTOMER Optimizer sprint-2 upwards into the Dev Ops workstream *without* having to type it in.

Assume that in CUSTOMER Optimizer, Sprint-2 the PAYMENT\_COLLECTION.ON\_INVOICE\_COLLECTION event is activated to publish the invoice XML to the YFS\_EXPORT table. The following additional configuration is entered into QA1 (*but not into QA2*).

The following components need to be created (in QA1).

1. Build service that writes XML into YFS\_EXPORT.
2. Create an action that executes this service.
3. Activate the PAYMENT\_COLLECTION.ON\_INVOICE\_COLLECTION event.

Now, that sprint 2 is over for CUSTOMER Optimizer (QA1), we want to merge the configuration up into Dev Ops (QA2) without having to re-type it.

To do this, we generate differences between

1. The current state of QA1 (end of sprint 1) as the *source*, and the CDT XML export created when Dev Ops was initialized from QA1 (CUSTOMER Optimizer) as the *target*.
2. This will give exactly the configuration that went into QA1 since Dev Ops kicked off.

The difference folder is shown below.



Sterling has created CDT files of JUST the configuration tables that have been affected in QA1.

If you look at the contents of the YFS\_FLOW XML, you will see that it has just one entry:

<?xml version="1.0" encoding="UTF-8"?>
<YFS\_FLOW NumRecords="1">
 <Insert DataLoadingService="" FlowGroupName="CUSTOMER"
 FlowKey="202104051414593901152" FlowName="writeToExport"
 FlowType="00" GraphUIKey="202104051414593901155" IsOutGoing="Y"
 IsPrintService="N" IsRealTime="N" OwnerKey="DEFAULT"
 ProcessTypeKey="GENERAL" SystemDefined="N"
 TransportTypeKey="LOCAL" \_\_ID\_\_="15"/>
</YFS\_FLOW>

The one entry in the file creates the new flow, WriteToExport, with an <Insert>.

Looking at the YFS\_EVENT XML, it looks like this:

<?xml version="1.0" encoding="UTF-8"?>
<YFS\_EVENT NumRecords="1">
 <Update ActionKey=" " ActiveFlag="Y" EnterpriseKey="DEFAULT"
 EventKey="ON\_INVOICE\_COLLECTION\_01"
 EventName="On Invoice Collection"
 Eventid="ON\_INVOICE\_COLLECTION"
 ExceptionType="ON\_INVOICE\_COLLECTION" ExtendEnterpriseFlag=" "
 ExtendSupplierFlag=" " ShipnodeKey=" "
 TransactionKey="PAYMENT\_COLLECTION" Version=" " \_\_ID\_\_="20">
 <OldValues ActiveFlag="N"/>
 </Update>
</YFS\_EVENT>

The one entry in the file updates the YFS\_EVENT table to activate PAYMENT\_COLLECTION.ON\_INVOICE\_COLLECTION.

Application of these differences to DevOps (QA2) will migrate the changes to DevOps without having to re-type.

Now we have merged CUSTOMER Optimizer (QA1), Sprint 2 into Dev Ops (QA2), let us begin work on Sprint 3 of CUSTOMER Optimizer.

Let us pretend that the only configuration we need to add to CUSTOMER Optimizer (QA1), Sprint 3 is a condition.

**Creating Condition Is B2B Order**.

1. Application Manager.
2. Application Platform.
3. Order Management.
4. Conditions.
5. Add new condition “Is B2B Order” as per the diagram below.



Now, that sprint 3 is over for CUSTOMER Optimizer (QA1), we want to push the configuration into Dev Ops (QA2). We do this as follows:

1. Generate full XML configuration export (QA1-3).
2. This CDT export is the *source* in the comparison. It represents the current config state of CUSTOMER Optimizer workstream at the end of sprint 3.
3. To determine the differences, use the CDT export from CUSTOMER Optimizer (QA1) the last time configuration was merged up into Dev Ops (QA3). This is CDT export Q1-2, and it is the *target* in the comparison.

The following difference files are created by the comparison.



You can see that only the YFS\_CONDITION table was affected in sprint 3 of CUSTOMER Optimizer, and that the contents of the file below is a single insert Insert for the new condition.

<?xml version="1.0" encoding="UTF-8"?>
<YFS\_CONDITION NumRecords="1">
 <Insert ConditionID="IS\_B2B" ConditionKey="202104051632593901272"
 ConditionName="IS B2B Order" ConditionType="PROCESS\_MODELLING"
 ConditionValue="OrderType = 'B2B'" GroupId="CUSTOMER"
 OwnerKey="DEFAULT" OwnerType=" "
 ProcessTypeKey="ORDER\_FULFILLMENT" ReturnType="" SeqNo="0"
 Type="STATIC" \_\_ID\_\_="2"/>
</YFS\_CONDITION>

The YFS\_ORGANIZATION XML is a “dummy” in that it actually does nothing. It just gives a count of the organizations whose configuration was affected.

<?xml version="1.0" encoding="UTF-8"?>
<YFS\_ORGANIZATION NumRecords="1"/>

Application of these differences to Dev Ops workstream (QA2) will port sprint 3 of CUSTOMER Optimizer (QA1) without having to re-type configuration into QA2.

**Use Case Two – CUSTOMER Optimizer (QA1) Goes Live**

In this use case, the lower workstream, CUSTOMER Optimizer (QA1), is put into production first. The example here shows how to use differences to move the configuration for QA1 into production.

To determine the differences, we generate differences as follows:

1. Pre-prod/prod is the *target*. The CDT export is Prod-0.
2. The *source* is the CDT export of CUSTOMER Optimizer (QA1) after the final sprint (sprint 3). This is the Q1-3 CDT export.

Comparing the two CDT exports generates the following files.



You can see that this is the combination of all sprints. This is applied to MC/pre-prod.

**Use Case Three – CUSTOMER Optimizer (QA1) Does Interim Go-Live After Sprint 1 And 2**

In this example an interim release of CUSTOMER Optimizer (QA1) goes live after sprint 2. Configuration for Sprint 3 has already been entered into QA1.

The difference configuration to apply to master configuration and pre-prod is generated as follows:

1. The CDT export at the end of sprint 2 (Q1-2) is the *source*.
2. Master configuration at the start of CDT Optimizer, Prod-0, is the *target*.
3. The differences give the configuration to put CUSTOMER Optimizer (QA1) live after sprint 2.



You can see that this excludes sprint 3 (the condition table is absent from the export). Applying these differences to master config/pre-prod will put CUSTOMER Optimizer (QA1) sprints 1 and 2 live.

**Use Case Three – Dev Ops (QA2) Going Live First**

In this use case, go-live CUSTOMER Optimizer is delayed, so Dev Ops (QA2) will be put into production first. All configuration related to CUSTOMER Optimizer (QA1) must be removed from QA2.

Again, we will use differences to accomplish this. We need to determine all the configuration currently in CUSTOMER Optimizer (QA1) that has been migrated to Dev Ops (QA2) and remove it from QA2.

1. The *source* for the difference generation is the CDT export at the start of CUSTOMER Optimizer (Q1-0).
2. As we have merged up to sprint 3 of CUSTOMER Optimizer (QA1) into Dev Ops (QA2), the *target* is the CDT export from CUSTOMER Optimizer (QA1) when sprint three was pushed up into Dev Ops. This is Q1-3.

Comparison of these two generates the following differences.



The files generated look almost identical to the ones when we were trying to migrate configuration from CUSTOMER Optimizer at the end of sprint 3 to Dev Ops. But examine the content.

<?xml version="1.0" encoding="UTF-8"?>
<YFS\_ACTION NumRecords="1">
 <Delete ActionKey="202104051418203901168"
 Actioncode="WRITE\_TO\_EXPORT" Actionname="Write to Export"
 Actiontype="USER" ComExtensionFlag=" " ComExtensionProgid=" "
 DbExtensionFlag=" " DbExtensionSpName=" " EmailAddress=" "
 EmailFlag=" " EmailTemplateFile=" " ExecuteFlag=" "
 ExecuteProgram=" " ExtendableCode=" " FaxFlag=" "
 FaxStylesheet=" " GroupId="CUSTOMER" HttpFlag=" " HttpUrl=" "
 InboxFlag=" " InboxListTemplate=" " InboxTemplate=" "
 InboxUserKey=" " InvokeFlowsFlag="Y" JavaExtensionFlag="N"
 JavaExtensionProgid=" " Lockid="1" OwnerKey="DEFAULT"
 OwnerType=" " PublishFlag=" " PublishReceivers=" " QueueKey=" "
 RealtimeFlag="Y" ReferenceKey=" " ReferenceType=" " \_\_ID\_\_="27">
 <OldValues/>
 </Delete>
</YFS\_ACTION>

This has generated a delete to REMOVE the action that was added as part of CUSTOMER Optmizer sprint 2.

Take a look at the YFS\_EVENT xml:

<?xml version="1.0" encoding="UTF-8"?>
<YFS\_EVENT NumRecords="1">
 <Update ActionKey=" " ActiveFlag="N" EnterpriseKey="DEFAULT"
 EventKey="ON\_INVOICE\_COLLECTION\_01"
 EventName="On Invoice Collection"
 Eventid="ON\_INVOICE\_COLLECTION"
 ExceptionType="ON\_INVOICE\_COLLECTION" ExtendEnterpriseFlag=" "
 ExtendSupplierFlag=" " ShipnodeKey=" "
 TransactionKey="PAYMENT\_COLLECTION" Version=" " \_\_ID\_\_="20">
 <OldValues ActiveFlag="Y"/>
 </Update>
</YFS\_EVENT>

It updates the PAYMENT\_COLLECTION.ON\_INVOICE\_COLLECTION event to deactivate the event which was added as part of CUSTOMER Optimizer sprint 2.

Application of this difference XML to the Dev Ops work stream (QA2) will remove all configuration belonging to CUSTOMER Optimizer allowing Dev Ops to go live first.