Product migration

COP - Coop (OmniPOS)

Exported on Apr 26, 2022

Table of Contents

1 Migration hints:

- Product guide / checklist for migration to a new product:
    - Product Upgrade (New Major Release)
- Check Master Migration Guide for useful migration hints for the version the project migrates to (and versions between old and new version).
    - Migration POS Client
    - Migration POS Server
    - Migration Store Device Control
    - Migration Launchpad
    - etc.

(language Thai). If something important is forgotten to be mentioned there, product should be informed.

- Check Unified maven builds in projects and use the parent pom corresponding to your product version. If the product version you are migrating to is not in the list, you are first and you have to create the parent pom in mod-pom-parent-projects.
- Get the old product version eg. 5.6.3-HF02-b01 and new product version eg. 5.9.1-b04 from these urls:
    - Central:
        - https://gitlab.gk.gk-software.com/ses/ses.git - SES (CIS, SDC, LPS, UCON, Portal, ENT-DB)
        - https://gitlab.gk.gk-software.com/pos-server/pos-server.git - POS-Server
        - https://gitlab.gk.gk-software.com/oms-mc/gk-launchpad.git - Launchpad
        - https://gitlab.gk.gk-software.com/backend/launchpad-apps - Launchpad-apps
        - https://gitlab.gk.gk-software.com/ses/lpa.git - LPA
        - https://gitlab.gk.gk-software.com/ses/storehub - StoreHub
    - OPOS:
        - https://gitlab.gk.gk-software.com/omnipos/omnipos.git - OPOS
    - SSC (only for migration in SSC development stream):
        - https://gitlab.gk.gk-software.com/system-management/wdm - WDM
        - https://gitlab.gk.gk-software.com/retail-services/self-scanning-service - Self-Scanning Service
- Use them for comparison in the chapters in Migration steps
- In mod-pom-parent-projects are listed only plugin versions. Compare versions of plugins in old and new product and change the version if necessary.
- Some classes, which are not located in these versions, can be found elsewhere in SVN/Git or you can take them from local m2 repository if necessary
- If you experience error:
    - omitted for conflict with during mvn clean install -Dsandbox -DskipTests: add the newer version to the dependency (or change existing version, possibly even remove conflicting dependency)

  - use mvn3 dependency:tree -Dverbose > tree.txt (verbose flag shows the dependency conflicts which are not present without it)
- Our first migration (from 5.6.3 to 5.9.1) can help you, check SVN for word "migration" or task COPM-1141)

1.1 How to migrate files:

1. Compare old_product vs new_product file - if no change, no need for file migration, otherwise step 2
2. Compare old_product vs project -> diff merge into new_product code -> use this merged code in the project

2 Migration steps:

Central should be migrated before OPOS.

2.1 POM.XML

- means main pom.xml of the application
    - central - https://gitlab.gk.gk-software.com/customer_solutions/projects/com.gk-software.cst.coo/blob/develop/central/pom.xml
    - OPOS- https://gitlab.gk.gk-software.com/customer_solutions/projects/com.gk-software.cst.coo/blob/develop/pos/pom.xml
    - WDM - https://gitlab.gk.gk-software.com/customer_solutions/projects/com.gk-software.cst.coo/blob/develop/wdm/pom.xml (SSC, different development stream)
    - self-scanning-service - https://gitlab.gk.gk-software.com/customer_solutions/projects/com.gk-software.cst.coo/blob/develop/self-scanning-service/pom.xml (SSC, different development stream, different lifecycle)
- Change version of mod-pom-parent-projects
- Change <version.dep.swee>, <version.dep.pos_server>, <version.dep.gkr.launchpad>, <version.dep.gkr.launchpad.apps> for central
- Change <version.dep.pos>, <version.dep.pos_server>, <version.dep.project_central> for OPOS

2.2 Eclipse

- First run from cmd: mvn clean install -Dsandbox -Dfast -DskipTests
    - Make sure you are using Maven version from Preparation of development environment (GIT)
- On all modules: Refresh → Maven Update Project (in case you worked before with SNAPSHOT of CENTRAL, check "Force Update of Snapshots/Releases") → Build all → Run
- The above steps should get Eclipse into working state with the migrated libraries

2.3 Compare and migrate:

2.3.1 POS

2.3.1.1 PROMO/PROMOJ

- Extended promo/promoj (_EXT)

2.3.1.2 XML LAYOUTS

- Check what was changed in product between versions, check if there is no conflict somewhere due to xml layout hierarchy - iFrames etc.

2.3.1.3 CSS for layouts

- Check what was changed in product between versions, check if some css change could affect current custom layouts

2.3.1.4 MODULES

- If module was included in new places in product, check if we are overriding it, if so, override on the new place if needed

2.3.1.5 DOIN CONFIGURATION FILES

- _EXT doin configuration files - instance Default may have some new required attributes, they must be copied from product
- _CST doin configuration files must be migrated too, because they often include copied Default instance from product

2.3.2 CENTRAL

2.3.2.1 UI5

- change versions of platform and ui5 in package.json in central\web-ui5\mod-pos-server-web-ui5-cst, central\web-ui5\mod-sdc-web-ui5-cst and central\web-ui5\mod-lpp-web-ui5-cst (correct versions can be found in Launchpad file build/war/package.json)
- delete yarn.lock files from central\web-ui5\mod-pos-server-web-ui5-cst, central\web-ui5\mod-sdc-web-ui5-cst and central\web-ui5\mod-lpp-web-ui5-cst and re-generate them by running mvn clean install on web-ui5
    - Warning: if this point is skipped, no error will probably occur until next release is performed; the release build is volatile to this

2.3.2.2 IMPORT MAPPING

- Generally, changed in Altova mappings need to be merged from product to the corresponding copied CST Altova mappings. This should be done via the Altova MapForce, in most cases it is not feasible to merge the .mfd file as XML.
    - As an exception, if there are changes in an Altova mapping in an import, which is in project fixed to an older version than the version in which changes were done in product, then these changes are not to be merged into project. The upgrades to such mappings are performed separately on-demand, not in scope of migration. It needs to be verified for every changed Altova mapping, if it is really this case.
- Java classes generated from Altova mappings are not completely deterministic/stable, so they can appear modified after re-generation. If there are changes in such classes in product without any change in the corresponding Altova mapping (.mfd), they can be safely ignored.

2.3.3 COMMON

2.3.3.1 JAVA CLASSES

- Check all classes with _EXT and merge the differences. Don't forget to check also private/protected methods, that has to be often copied in delegate classes, don't check only overriding methods. If you find missing comments (// CST ---, // EXT ---, // COPY ---) at the delegate patterns which should be used according to conventions in Implementation, please add them to distinguish product changes from project changes.
- Similarly, check all classes with _CPY / _BRK. Merge all differences from product into _CPY, which has to be as close to the product class as possible (i.e. a 1:1 copy of the class only with changed name, logger, perhaps visibility of some methods, etc.). The _BRK class extends either a _CPY class or the product class directly and its instance is used to replace the product bean in Spring / component framework. So relevant changes in product need to be merged to _BRK classes similarly to _EXT classes. Comments for distinguishing project changes from product changes should also be used and added if missing. Do not forget to update or add information to a comment in these classes, which version of product there were last merged with.

2.3.3.2 SPRING AND OTHER XML files

- Overlapped xml files (in src/main/resources) like bootstrap.xml, md-replication.xml, web.xml etc.
- Extended xml files (pos-client-group-ext.xml, pos-service-group-ext.xml)

2.3.3.3 TEMPLATES AND PATCHES

- Merge changes from product into template files overwritten in project.
- Patches for product property files - if we patch some numbered record (from list), we must check the order was not broken and that we still patch the correct record (inputDomains.properties, hal.properties etc.), product unfortunately don't always add new props to the end, what was in our patch .26 can now be easily .30
- Central:
    - msg-router.properties (POS-Server) and transaction-pool.properties (SDC) and vice versa - the process groups must be synchronized across systems (e.g. 201 was removed and 207 added on SDC).

2.3.3.4 TESTS

- Try to run unit tests on all projects - if some of them fail, fix them
- Run integration tests (use Maven command from Jenkins job COO.MAIN.07-VERIFY).

2.3.3.5 DATABASE

- Takeover changes in the DBM modules, if there are any overrides in project DBM
- Update DB structure
    - Central
        - run "build hdb update" in trunk\build\dbm\dbm-sandbox_sandbox\data-master
        - run "build hdb update" in trunk\build\dbm\dbm-sandbox_sandbox\data-tenant
    - OPOS
        - set correct connection settings to your sandbox derby db in trunk\build\pos-full\pos-full-sandbox_sandbox\data\md\data\derby\build.properties
        - run "build derby update" in trunk\build\pos-full\pos-full-sandbox_sandbox\data\md\data
- Update referential MD database for OPOS (ref-db.zip)
    - either make both Central and OPOS runnable, or have the OPOS developer connected to the Central developer
        - then let the OPOS download the MD database
        - or create a pump, let the OPOS download the pump and filled its MD from the pump
    - or apply the differential changes manually to the existing OPOS MD database filled with data, if feasible
    - or after migration is finished and snapshot is created, the database can be created and filled by QA
- Check replication is working together with Central developer
- When QA first runs application after the migration and fills their db with data, ask that db from them and commit it as a testing db into project (trunk\build\pos-full\pos-full-sandbox\src\main\refdata\md.zip)

- Check If was added some new table into the tx-pool schema and if is needed create new version of script delete_store_transactions.sql and add this table into it.

2.3.3.6 VERSION DESCRIPTOR

- If you added some new configuration in project, do not forget to add this to a version_descriptor.xml, so it is correctly downloaded for real environment from Store Manager

2.3.3.7 INSTALLER, CDPL, SCA...

- Check installer, cdpl, sca changes, merge them - check ALL files here including pom.xml

2.3.3.8 DATA CONTAINERS

- Compare xsd folder in project and product data containers and merge changes
- If xsd was changed in an incompatible way (it should not happen but...) then xml with data must be also updated
- New xsds in product data containers should also be merged, including the corresponding changes in catalog.xml

2.4 Additional steps after migration:

2.4.1 ISSUES PENDING ON PRODUCT

2.4.1.1 BUGS FIXED IN PRODUCT

- Go through pending bugs in project with Bug Class = Product Error and check whether any of them were fixed in the product version the project migrated to.
- Link found issues as children under the migration issue ("has to be done after") and make sure they will be resolved.

2.4.1.2 TECHNICAL DEBT

- Go through Technical debt and search for any technical debt record which can be resolved after the migration. Typical usecase is removal of workarounds in project which are not needed anymore, e.g. because needed behavior was implemented or missing extensibility was introduced.
- Do not forget to mark the resolved records green.

2.4.2 RELEASE

2.4.2.1 QA

- Make sure to inform QA about any relevant breaking changes or anything nonstandard they have to take into account
    - e.g. necessity of performing DB complete rebuild after migration