Oracle® Communications Launch Cloud Service

Integration Guide

Release 26B
G47971-01
April 2026

ORACLE

Oracle Communications Launch Cloud Service Integration Guide, Release 26B

G47971-01

Contents

About This Content

This document describes the pre-built integration between Launch and Siebel CRM.

Audience

This guide is intended for administrators who are familiar with Oracle Communications Launch Cloud Service and Siebel CRM applications.

Conventions

The following text conventions are used in this document.

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Integration with Siebel (CRM)

The pre-built integration between Launch and Siebel CRM allows you to centrally manage product and services portfolio across your ecosystem. Launch serves as the primary source for catalog definitions, with Siebel CRM consuming these definitions. By using the pre-built integration between the two applications, you can perform a one-time migration of catalog definitions from Siebel CRM to establish them as the authoritative source in Launch.

You can innovate faster by centrally managing your product portfolio for your entire eco-system, including Siebel CRM.

Once Launch is baselined with the catalog definitions, any new products or changes to existing ones are handled in Launch with automated distribution to Siebel CRM.

Related Guides

Table 1-1 contains information about other useful sources of information for the integration process.

Table 1-1 Related Guides

Reference	Description
Launch Cloud Service User's Guide	Describes how you can create, publish, and manage product offers.
REST API Reference for Launch Cloud Service	Provides the REST API reference document for Launch Cloud Service.
Implement CX Industries Framework	Describes the setup and implementation of the CX Industries Framework required to deploy Launch Cloud Service.
Siebel Update 21.12: Siebel CRM 2021 TOI: Product Administration APIs for Siebel Customer Order Management Functional Overview Siebel Update 22.5: Siebel CRM 2022 TOI: Pricing Administration Rest APIs for Siebel Customer Order Management Functional Overview Siebel CRM 25.5 Update Guide and Known Issues	Describes the changes in the Siebel REST APIs supporting the integration.

Supported Versions

- Launch version 26.04 or later
- Siebel CRM version 26.1 or later

Supported Integration and Mapping

Table 1-2 lists the entities you can do a one-time migration to and publish. The table also lists the predefined mappings available to you in the JSON mapping file.

> **Note**
>
> Ensure that migration upgrade and downgrade rules exist between promotions in the same Organization/Business Unit.

Table 1-2 Supported Integration and Mapping

Siebel Entity	Launch Entity	What can you sync?
Catalog	Catalog	Definition Category Association
Category	Category	Definition Sub-categories Product Association
Product Class	Product Specification	Definition Parent-Child Relationships
Attributes	Attributes	Attributes Smart Part Number (user defined)
Products (Simple and Customizable products)	Simple Offer, Bundle Offer	Definitions Category/Class/Product Line association (Catalog needs to be migrated before migrating Product) Prices and Adjustments – simple and Customizable products Volume Discounts for Simple products Compatibility and Eligibility Rules and Recommendation Discount Products (products with negative price) Customizable Product level price and Constraint Rules (six templates supported) Attribute Adjustment Discount Matrix Volume Discount
Product Line	Product Line	Definition Product association Compatibility Rules
Price list	Price list	Definition and its price list items Attribute Adjustments Aggregate discounts Volume discounts

Table 1-2 (Cont.) Supported Integration and Mapping

Siebel Entity	Launch Entity	What can you sync?
Promotion	Package	Definitions Category/Class/Product Line association (Catalog needs to be migrated before migrating Promotion) Components along with Aggregates and Option Class overrides Eligibility, Compatibility and Upgrade/downgrade Rules Commitment Terms Siebel Promotion Price Overrides Siebel Promotion Domain Product Overrides Siebel Promotion Commitment Terms Overrides Attribute Adjustment Discount Matrix Volume Discount

Setting Up Launch Siebel CRM Integration

The following table lists the setup tasks you need to perform in Launch, Industry Framework, and Siebel for one-time migration from Siebel to Launch and subsequent publishing from Launch to Siebel.

Prerequisites

Table 1-3 lists the prerequisites you need to perform in Launch, Industry Framework, and Siebel for one-time migration from Siebel to Launch and subsequent publishing from Launch to Siebel.

Table 1-3 Prerequisites

No.	Application	Task	Mandatory?	Description
1.	Industry Framework	Create Integration User	Yes	This is required to facilitate the integration between the two applications.
2.	Industry Framework	Configure the downstream systems	Yes	This is required to ensure to configure the downstream system instance for receiving publishing events.

Table 1-3 (Cont.) Prerequisites

No.	Application	Task	Mandatory?	Description
3.	Launch	Register destinations	Yes	This is required to configure the right downstream system instance to receive the publishing events.
4.	Launch	Configure Entity Profile	Yes	This is required to ensure that Launch can model catalog definitions based on Siebel.
5.	Launch	Setup Default entities	Yes	This is required to meet the Launch requirement of Products and Services to have specifications.
6.	Siebel	Create client credentials and security requirements	Yes	N/A
7.	Siebel	Integrate Launch with third party content management system	No	This is required only if images of devices, sales collaterals are associated to product definitions in Siebel.
8.	Siebel	Identify the integrations to Product definitions that is required for integration	Yes	This is required to baseline Launch with the product definitions that are required for the one-time migration.
9.	Siebel	Add the value Discount in Type dropdown loads under Administration - Product > Products tab	Yes	This is required to publish the Discount type offer correctly to Siebel.
10.	Siebel	Validate Product definitions data	Yes	This is required to ensure the working of one-time migration and publishing.

Table 1-3 (Cont.) Prerequisites

No.	Application	Task	Mandatory?	Description
11.	Launch	Create the Organization in Launch (to support Siebel multi-org setups for Products and Pricelists)	No	You can create your own organization in Launch. If you want to publish products with such organization, you need to create the organization first in Siebel and in Launch.
12.	Siebel	Create the Organization in Siebel	No	You can create your own organization in Launch. If you want to publish products with such organization, you need to create the organization first in Siebel and in Launch.

Setup Task Details

To set up Siebel CRM and Launch for catalog migration and publishing, follow these steps:

1. Create a new user on the security console with the user name **FABRIC_SYSTEM_USER** and the role **Communications Catalog Administrator**. If you have already created this as part of the Launch setup, no additional setup is required.
2. Configure your external application in the Industry Framework for each instance. For more information, refer to the topic *Integrate External Applications* to add a Spoke End Point in the article *Implement CX Industries Framework*, on My Oracle Support, Doc ID 2720527.1.
3. Register **Destination** in Launch for publishing to the Siebel CRM instance. There are multiple Siebel CRM instances such as Development, SIT, UAT, and Production, but you need to set up each instance individually for Launch in order to publish to the correct Siebel instance.

   Configure destinations in Launch to appropriate life cycle status for which you would like to publish to external application. For example, you can configure a destination for test instance to the **Ready to Publish** life cycle status and configure one for production instance to the **Active** status.

   While adding destinations, use

   a. Name - The target pre-selection key from the Connection Descriptor in the Industry Framework.
   b. Type - The target name used in the Launch Catalog Sync configuration.
   c. Internal - Set this to ON.

   To create a Siebel CRM destination:

a. Navigate to the Launch user interface, click the **Administration** tab, and then click **Lifecycle status**.

b. Select the new lifecycle configuration version in the **PENDING** state.

c. Choose the Edit option for the In-design state and click **Add Destination.**

d. Provide the Name, Type, and Publish Sequence. Click **Add.**

> **Note**
>
> Destination Name and Type should match the spoke system configuration in the CX Industries Framework.

e. Select the Siebel destination name in the Destinations field and save it.

f. Choose the **Activate** option for the **PENDING** life cycle configuration version.

g. Verify that the new lifecycle configuration version changed to the **ACTIVE** state.

For more information on how to configure the Destination in Launch, see *REST API Reference for Launch Cloud Service* and Publish Catalogs in *Launch Cloud Service Implementation Guide*.

> **Note**
>
> The destination Name setup in Launch should be an exact match to the system descriptor creation using config.ms in Customer Experience Industry Framework (CXIF). This helps in identifying the correct Siebel instance for one-time migration (“Source” field in migration job) and in the mapping for Publishing to the appropriate Siebel instance. Ensure that you use the same name for system parameter name in system Descriptors while applying the configuration in step 5 below.

4. Configure the Entity Profile to ensure that Siebel CRM supported product modeling is followed by Launch using the Entity Profile tile in Administration space.

   No two applications are the same when it comes to modeling capabilities and so is the case between Siebel CRM and Launch. Though the result might be the same, the constructs might be different between the applications, and while integrating the applications, you need to factor in any restrictions of the target(spoke) application to ensure an error-free publishing of catalog definitions. Some of the common patterns between Siebel CRM and Launch can be classified as:

   a. Both the applications having the same construct of capabilities and restrictions for an entity at par - for example, Catalog, Category, Product line entities, and so on.

   b. Both the applications having minimal common capabilities and/or additional capabilities or restrictions in one and not in the other - for example, Siebel CRM and Launch can have commitment terms but Launch has the provision to configure multiple commitment terms to an offer, while only one commitment term is supported in Siebel CRM promotions.

   Let's say we migrate the catalog definitions from Siebel CRM to establish them as the primary source in Launch.

   Now, every simple product definition of Siebel CRM will have only one price type associated with it - one-time or recurring. Launch supports multiple price types to be associated to a single simple offer including support for usage type price and alterations, price based on relative effectiveness, and so on.

Should the migrated offer be enriched or revised to include any of the non-Siebel CRM supported constructs, the publish to Siebel CRM will fail.

So, when we migrate catalog definitions from Siebel CRM to establish them as the authoritative source in Launch and make changes to those entities, we need to ensure that publishing to Siebel CRM (being the order capture application) does not fail. This also applies to catalog definitions that are created in Launch. Hence, the need to have certain restrictions to be applied in Launch in line with Siebel CRM capabilities to have a successful publishing.

5. Set default entities for the Siebel entities in Launch for meeting the minimum requirement of integration. As part of the migration process, the following mandatory resources are created as a default specification in Launch if migrated entities do not contain the same as part of the association.

    a. Every Siebel simple product definition in Launch requires a product specification association in Launch regardless of whether it is of Device, Accessory, or Service type. In addition, every simple product of type requires a Service Specification association. Since Siebel allows you to create simple products without a class association and there is no notion of a Service Specification (CFS Spec), while doing the one-time migration, default entities are appended to the simple product in Launch.

        For sample payload, see **Service Specification** and **Product Specification** in Appendix.

    b. For migrating and publishing Siebel Aggregate discounts, you need to seed Launch with a custom profile specification having a criteria parameter – product offers and quantity.

        For sample payload, see **Siebel Default Aggregate Discount CPS** in Appendix.

    c. For migrating and publishing Siebel Eligibility rule, you need to add the following parameters using Common Business Configuration.

        i. Country

        ii. State

        iii. City

        iv. Post Code From

        v. Post Code To

        For more information on adding lookup values using the extensibility framework, see *Launch Cloud Service Implementation Guide*.

    d. Siebel CRM Attribute Adjustments are made available in Launch as Attribute based adjustments (ABP). Launch supports attributes from Product Specifications, Service Specifications, and Custom Profile Specification characteristics to support Attribute based adjustments. Such definitions are published to Siebel CRM as Product Based Adjustments Discount Matrix type. Since Siebel CRM does not support Service Specification based ABP, use Product Specifications and Custom Profile Specifications to migrate the Discount matrix from Siebel CRM and subsequent publishing to Siebel CRM.

        For sample payload, see **Siebel Discount Matrix CPS** in Appendix.

    e. The Launch migration rule penalty requires a Pricelist association to display penalty details accurately on the UI. However, Siebel does not have any Pricelist information for migration rule penalty. During the migration from Siebel to Launch, the Siebel-Default-PL is associated with the penalty POP to ensure the Launch UI can present the data correctly. This default Pricelist, Siebel-Default-PL, is only required for the migration rule penalty POP.

For sample payload, see **Siebel-Default-PL** in Appendix.

> **Note**
>
> Ensure that the default entities for migration is created before initiating any migration job.

6. Create client credentials and security requirements for Siebel CRM. See Overview of Using the Siebel REST API and Using Siebel REST API For Siebel Telco in the *Siebel REST API Guide*.
7. Integrate Launch with third party content management system. This is an optional task and you require it if you want to migrate and publish content between Launch and Siebel CRM.

   To view the literature and images in Launch after the migration, associate the Siebel CRM products using URL-based Literature records into content store. For Siebel CRM products that are associated with File-based Literature, move them to content store and create the URL-based Literature records. For more information on creating URL based Literature in Siebel CRM, see the chapter **Literature** in the *Siebel Applications Administration Guide*. You can find the required version of Siebel CRM documentation from the Oracle Siebel CRM Documentation page.
8. Identify extensions on product definitions that need to be part of the integration. This is necessary to plan for extensions to be made in Launch before you proceed with the migration. For more information on how to extend Launch application, see Configure and Extend Launch in *Launch Cloud Service Implementation Guide*.

   > **Note**
   >
   > Any changes to the mapping files related to both migration and publishing, due to the extensions, must be redone after every product release upgrade. This is a manual activity that needs to be factored in your project implementation.
9. Launch simple offer with DISCOUNT product type gets published to Siebel with Type as Discount. Siebel OOB doesn't provide Discount value in Type drop-down. So, make sure to add Discount value in Administration - Product > Products tab > Type drop-down before initiating publish of Discount offer to Siebel CRM.
10. Validate before you initiate the one-time migration of product definitions:
    a. Ensure that the simple product have the right type to map it to the Product Offering types in Launch (Device, Accessory, or Service). See the topic, *Product Type and Service Type* in this article.

       The migration process uses the rules defined in the grammar file to decide the Launch product types. With OOB rules, simple products having one-time price will be migrated as device and those with recurring price will be migrated as service. However, the integration process allows you to change the OOB rules by using extensions.

       You need to pass a header X-User-Product-Type with value true, then classification of simple product types would be based on 'Type' drop-down field loads under Siebel UI Administration - Product > Products tab.

       For example, to use Product Type or Service type as the fields to identify a Product as a device or service,

       - Ensure that all the simple products that you want to migrate have the Product Type or Service type set correctly.

- Update the default mapping rule to decide the Launch product type based on the values set in the Product Type or Service Type or both. For more details on how to handle extensions, see Handling Siebel CRM extensions.

b. Discount offers in Siebel are with negative amount that is mapped to Launch ProductOffering of type “DISCOUNT”. During migration, such discount offer pricing is set to 0 and a fixed discount with amount value provided in Siebel is created in Launch. Though the display shows negative amount in the Launch UI, while creating such Discount Type offers and publishing to Siebel they are converted to negative amount in Siebel. If Launch doesn't have product offering price start date, then during transformation to Siebel pricelist item, the Product Offer start date is used.

c. Ensure that the product definitions have an effective start date. This is required in Launch because it determines the validity of associations to other entities. Also ensure that the effective dates of attributes associated with a product class are valid and do not extend beyond the product class’s effective dates.

d. All the versioned definitions of Siebel, that get published from Launch will set the released flag to true.

e. Any attribute with its domain type defined as “Enumerated” should have at least one value.

f. Customizable Products must have at least one component to migrate to Launch, otherwise the migration job will fail while creating bundle offers without components in Launch.

g. Ensure that attribute sequences are not duplicated.

h. Ensure that every simple product has a price list item, even if it has a value of zero.

i. Ensure to turn on the Orderable flag while creating a product in Launch. This is required to make the product available in drop-down lists and dialog boxes in Siebel post publish.

j. Constraint Rules with below templates only are supported in Launch and hence migration job ignores any rules using unsupported templates.

- Require
- Require (mutual)
- Exclude
- Constrain attribute conditions
- Constrain attribute value
- Constrain relationship quantity

One-Time Migration

You migrate entities from Siebel to Launch using a migration job in Launch in the same way as any import or export job. For more information, see Create Catalog Entities.

> **Note**
>
> This is a one-time migration and not to be used as a sync.

The one-time migration of Siebel CRM entities is used to onboard a Siebel customer by baselining Launch with the Siebel Catalog definitions. Any further changes to the migrated

catalog definitions in Launch and/or any new Catalog definitions are originated in Launch and distributed to Siebel CRM using the Publish process.

Migration Job Parameters

You initiate a migration job in Launch using the **Job Management** tile's **External Jobs** in **Administration**.

To create an external job for migration, you need to provide the following information:

- **Name**: A name for the migration job.
- **Source**: The name of the source configured in Catalog Sync. The migration job identifies the rules for migration based on this. The out-of-the-box configuration uses the name 'siebel'.
- **API**: The migration API to be run. See Migration API for more information about the available API calls.
- **Query Parameters**: The migration API uses query parameter values to fetch the entities from Siebel CRM.
- **Header Parameters**: The header parameter is optional. However, if the header parameter is provided in the request, the associated required parameter must also be included, as specified in the table below. For more information, see Table 1-6.

Migration Paths

Below are the available set of migration paths to bring entities from Siebel to Launch.

Table 1-4 contains the complete list of supported migration paths.

Table 1-4 Supported Migration Paths

Siebel API	What gets migrated
/migration/catalog	Catalog and its associated categories
/migration/attribute	Top-level attributes
/migration/class	Product Class, its Attributes, Hierarchy, and Smart Part Num
/migration/pricelists	All Pricelists
/migration/productLine	Product line
/migration/package	Promotions will be migrated, but all components and references must be migrated separately.
/migration/packageWithDependencies	Promotions will be migrated with all of their dependencies.
/migration/simpleOffering	Products will be migrated as atomic product offerings without their references. The standard classification of simple product will be based on price Type.
/migration/simpleOfferingWithDependencies	Products will be migrated as atomic product offerings along with their references. The standard classification of simple product will be based on price Type.

Table 1-4 (Cont.) Supported Migration Paths

Siebel API	What gets migrated
/migration/bundle	Products will be migrated as bundle products without their references. Products will be migrated as Commercial Bundles by default.
/migration/bundleWithDependencies	Products will be migrated as bundle products with their references. Products will be migrated as Commercial Bundles by default.
/migration/compatibilityAndMigrationRules	Compatibility and migration rules associated with the Promotion and its components.
/migration/productRecommendation	Product Recommendation Rule associated with the Product
/migration/ productCompatibilityRule	Compatibility rules associated with the product
/migration/aggregateDiscounts	Aggregate Discounts
/migration/entitlements	Entitlement Templates
/migration/packageWithDependenciesBulk	Siebel Promotions migration in bulk

All API calls support migration of the Siebel CRM catalog entities using three options:

- Name match
- ID match
- Siebel searchSpec Expression: Helps us to create complex SQL queries to retrieve data. For more information, see Siebel documentation.

Table 1-5 contains examples of migration path calls with sample query parameters and their values.

Table 1-5 Migration Paths

Scenario	Siebel API	Query Parameter (Example)	Query Value
To migrate a particular Catalog entity	/migration/catalog	$.SiebelMessage['ListOf Base Catalog Admin'] ['Product Catalog'].Name	Catalog Name
To migrate a Product Class using its ID	/migration/class	$.SiebelMessage.ListOf SWIAdminISSClassDefi niti onIO.['SWI ISS Class VOD BusComp'].['VOD Id']	Product Class ID
To migrate a set of Product line entities	/migration/product Line	$.SiebelMessage. ['ListOfSWI Admin Product Line'].['Admin Product Line'].searchspec	[Name] LIKE ‘ProductLine Name1* OR [Name] LIKE ‘Name2*'

> **Note**
>
> Siebel migration use templates to query to Siebel APIs which require the request body. The request body is passed using templates. For more information, see Supported Templates. You can change the Query parameters in the template file. For example, the productPOSTTemplate.json in the simpleOfferingWithDependencies API uses the searchspec to identify the product to be migrated. This can be changed to ID, Name, or any other field as required. searchspec can be a wildcard search and supports multiple fields.

For more information on query payload and search specification, see the Siebel REST API Guide. You can find the required version of Siebel REST API documentation on the Oracle Siebel CRM Documentation page.

The templates can be updated using the Update Template File API. You can update the templates using the Update Template File API. For more information, see Update Template File in this guide.

Table 1-6 contains the list of header parameters for the migration job:

Table 1-6 Header Parameters

Key	Required?	Description
X-Source-System	yes	Name of the source system; auto populated based on Source being selected
X-Destination-System	yes	Name of the Destination system; auto populated to Launch
X-Source-PreSelection	yes	The target pre-selection key configured in the Connection Descriptor in Industry Framework, pointing to the Siebel instance from which Catalog definitions needs to be migrated.
X-User-Project-Name	no	If not provided, the migration job ID will be used for the Initiative Name.
X-User-Project-Id	no	If not provided, the migration job ID will be used for the Initiative ID.

Table 1-6 (Cont.) Header Parameters

Key	Required?	Description
X-User-Skip-Target-Spec	no	This header is used to skip the processing of the TRANSFORMATION and LOAD phases of a target request specification. It accepts a comma-separated list of values which can either be componentName or name field of the target_request_spec defined in the mapper file. For example, to skip the constraints rule, add this header in new migration job page during creation of new job by clicking **Add Header Parameter** button available on Launch UI and pass value as ConstraintRules. This will skip the processing of all the constraints rule for a CP without turning them off manually on Siebel UI. For more information, see Table 1-7.
X-Version	no	Used to pick the correct version of the mapper file. If not provided, mapper file present in default folder in content store or seeded one is used.

> **Note**
>
> Ensure that you do not skip the dependent entities by using the X-User-Skip-Target-Spec header as this might cause the migration to fail. For example, you cannot provide Attribute as a value to this header to skip them, as a Product Specification may be referring to these attributes. Similarly, you cannot input ProductSpecification as a value to this header to skip it, as some Product Offerings could be using it. Proceeding with these actions will result in a failed migration job. For more information on the values that can be passed to this header, see Table 1-8.

Table 1-7 Use Cases for the X-User-Skip-Target-Spec Parameter

Use Case	X-User-Skip-Target-Spec Header Value	Description
Skip the constraint rules	ConstraintRules	Skip the constraint rules defined on CP.
Skip the eligibility rules	ProductEligibilityRules	Skip all the eligibility rules defined on Simple Product, CP, and Promotion.
Skip only the Simple Product eligibility rules	SimpleProductEligibilityRules	Skip all the eligibility rules defined on Simple Product alone.

Table 1-7 (Cont.) Use Cases for the X-User-Skip-Target-Spec Parameter

Use Case	X-User-Skip-Target-Spec Header Value	Description
Skip only CP eligibility rules	CpProductEligibilityRules	Skip all the eligibility rules defined on CP alone.
Skip only Siebel Promotion eligibility rules	PromotionEligibilityRules	Skip all the eligibility rules defined on Siebel Promotion alone.
Skip the constraints and eligibility rules	ConstraintRules, ProductEligibilityRules	Skip all the constraint rules defined on CP along with any eligibility rule defined on Simple Product, CP, and Siebel Promotion.
Skip the constraints and CP eligibility rules	ConstraintRules, CpProductEligibilityRules	Skip the constraint and eligibility rules defined on CP.

Table 1-8 X-User-Skip-Target-Spec Header Values

X-User-Skip-Target-Spec Header Value	Description
Attribute	Skip both parent and child product specification attribute
ChildProductSpecAttributes	Skip the child product specification attribute
ParentProductSpecAttributes	Skip the parent product specification attribute
ProductSpecification	Skip both parent and child product specification
ParentProductSpecification	Skip the parent product specification
ChildProductSpecification	Skip the child product specification
Productline	Skip the product line
Pricelist	Skip the price list
ConstraintRules	Skip the constraint rule
ProductEligibilityRules	Skip the eligibility rules for Simple product, CP, and promotion
SimpleProductEligibilityRules	Skip the Simple Product eligibility rules
CpProductEligibilityRules	Skip the Customizable Product eligibility rules
PromotionEligibilityRules	Skip the Promotion eligibility rules
ProductOfferingSimple	Skip the Simple Products
ProductOfferingBundle	Skip the Customizable product
Package	Skip the Siebel promotion
ProductUpgradeRules	Skip the upgrade rules
ProductDowngradeRules	Skip the downgrade rules
ProductCompRules	Skip the compatibility rules
ProductRecommendationRules	Skip the recommendation rules

Migration Process

The Launch-to-Siebel CRM integration supports two patterns of running the one-time migration job:

1. Top-down migration: You can migrate a single promotion or a set of promotions and all their references to the leaf level products, all product classes, and rules at the same time.

2. Bottom-up migration: You can migrate entity by entity, starting with the product class and its attributes and smart part number, followed by product definitions (even simple products first followed by customizable products), and promotions.

> **Note**
>
> - Every simple product definition must have a price list item, even if it is zero.
> - Every customizable product must have at least one domain relationship.
> - Ensure the effective dates of attributes are earlier than the effective dates of the associated product class.
> - Ensure the attribute value display sequence uses unique values.

Publishing Process

When you publish an initiative, all the entities in the initiative are pushed to Siebel CRM. For more information on how you can publish your catalog entities to the spoke systems, see Publish Catalog Entities in *Launch Implementation Guide*.

Publishing will release all the Siebel versioned objects during the publish process for the below versioned objects. See "Publish Mapping File" in Siebel Extensions Handling:

- Product Class
- Attributes
- Product Definitions
- Promotions

> **Note**
>
> - Cardinal Rules of Integration:
>   - Any entity and its properties published by Launch must not be enriched in Siebel.
>   - Any entity not published from Launch is allowed to be enriched in Siebel.
> - While publishing the package to a Siebel promotion, a zero-value price list item will be added in the following scenarios:
>   - The package offer in Launch has an end date defined.
>   - The package offer in Launch has an E&C rule defined.

Enrichment after Publishing

To enrich objects in Siebel CRM after they have been published in Launch, you need to lock the versions in Siebel, enrich them, and then release them as a new version. You should do this **only** for entities that are not supported by the Launch-to-Siebel CRM publishing framework.

A few of the entities that can be enriched are:

- Product Class

  - User Interface
  - Linked Items
  - Resources
  - Scripts
  - Display Name
  - Properties
  - Constraints
- Product
  - User Interface
  - Linked Items
  - Resources
  - Scripts
  - Display Name
  - Properties
  - Constraint Rules (Only types that are not published by Launch. Refer 10.j.)
- Promotion
  - Merge
  - Split
  - User Interface

2

Integration with PDC (BRM)

This chapter outlines the configuration steps required to integrate Launch with Pricing Design Center (PDC) or Billing Revenue Management (BRM). The integration helps customers who want to take advantage of the latest capabilities of the Oracle Launch Cloud Service while leveraging their existing investment in BRM.

Related Guides

Table 2-1 contains information about other useful sources of information for the integration process.

Table 2-1 Related Guides

Reference	Description
Launch Cloud Service User's Guide	Describes how you can create, publish, and manage product offers.
REST API Reference for Launch Cloud Service	Provides the REST API reference document for Launch Cloud Service.
PDC REST Services Manager Overview	PDC documentation to create client credentials and security requirements for PDC/RSM deployment.

Supported Versions

The minimum required application for this feature is:

- Launch release version 26.04 or later
- Oracle PDC/BRM 12 PS8 plus Patch 35361657

Supported Integration and Mapping

Launch-PDC Integration uses the mapping service which enables you to create a proxy API that can push the data into PDC/BRM. The mapping service currently works for the following entities in Launch. All other entities are ignored. Table 2-2 lists the entities that can currently be mapped.

Table 2-2 Supported Integration and Mapping

Launch Entity	PDC Entity	What can you synchronize?
Simple product offering	Charge offer of Subscription type	Definition, pricing, charging terms
Simple product offering with fees and alterations	Charge offer of Subscription type / Item / Account type (based on Launch definition).	Definition Pricing and Adjustments

Table 2-2 (Cont.) Supported Integration and Mapping

Launch Entity	PDC Entity	What can you synchronize?
N/A	Discount offer of Subscription type. The name of the discount offer will be post fixed with _DISCOUNT. For Digital Business Experience (DBE) customers, the _DISCOUNT post fix won't be there. Discount offers will be post fixed with _DISCOUNT only for NON-DBE customers.	Charging terms
Simple offering of device/ accessory type	Charge offering of Account type	Definition Pricing
Attribute based pricing	Charge Selector	Definition Pricing
Package	Package	Definition Components Commitment terms
Service Bundle	Bundle	Definition Components Commitment terms
Attribute based adjustment	Discount Selector	Definition Pricing
Attribute Based Pricing with Custom Analyzer Rule	Generic Selector	Definition Pricing
Simple Product Offering with Trigger Conditions (except Impact Category, Price Plan Name, Currency) are configured	Trigger Spec	Definition Pricing
Simple Product Offering with Trigger Conditions such as Impact Category, Price Plan Name, and Currency are configured	Charge Selector Spec	Definition Pricing
Simple Product Offering with Roll Over Configuration	Rollover Rate Plan	Definition Pricing
Simple Product Offering with Charge Share	Charge Share Offer (Distribution Rate Plan and Distribution Offering)	Definition Pricing

Setting up Launch PDC/BRM Integration

There are a few setups required to be done in Launch, Industry Framework and BRM for Publish from Launch-to-PDC.

Prerequisites

Table 2-3 Prerequisites

No.	Application	Task	Mandatory?	Description
1.	Industry Framework	Create Integration User	Yes	This is required to facilitate the integration between the two applications.
2.	Industry Framework	Configure the downstream systems	Yes	This is required to ensure to configure the downstream system instance for receiving publishing events.
3.	Launch	Register destinations	Yes	This is required to configure the right downstream system instance to receive the publishing events.
4.	Launch	Configure Entity Profile	Yes	This is required to ensure that Launch can model catalog definitions based on PDC/BRM.
5.	PDC REST Services Manager	Create client credentials and security requirements	Yes	N/A
6.	Launch and PDC/BRM	Set up Configuration entities	Yes	Configures the set up entities between Launch and PDC.

Setup Task Details

1. Create a new resource user with the user name **FABRIC_SYSTEM_USER** and the role **Communications Catalog Administrator** using the Security Console. If you have already created this as a part of the Launch setup, no additional setup is required.
2. Configure your external application in the Industry Framework for each instance. For more information, refer to the topic Integrate External Applications to add a Spoke End Point in the article *Implement CX Industries Framework*, on My Oracle Support, Doc ID 2720527.1.
3. Register Destination in Launch for publishing to PDC/BRM instance. Usually, you would have many PDC/BRM instances such as (Development, SIT, UAT and Production). Each instance of BRM is a destination that needs to be setup for Launch to publish to the correct PDC instance.
4. Configure the Entity Profile to ensure that PDC/BRM supported product modeling is followed by Launch using the Entity Profile tile in Administration space.

   No two applications are the same when it comes to modeling capabilities and so is the case between PDC/BRM and Launch. Though the result might be the same, the constructs might be different between the applications, and while integrating the applications, you need to factor in any restrictions of the target(spoke) application to ensure an error-free

publishing of catalog definitions. Some of the common patterns between PDC and Launch can be classified as:

5. Create client credentials and security requirements for PDC/BRM. See PDC/BRM documentation for information about creating client credentials and security requirements for PDC/RSM deployment.
6. Ensure that the services, events and service-event maps, general ledger IDs (GLID), tax codes, balance elements, custom analyzer rules, value maps, standard zones, price tags, impact categories, Policy Specifications, and Time Model required for charge or discount offers. Use the PDC Documentation for setting up the entities in PDC and use the Launch API documentation for setting up the same in Launch.

   Before setting up integration, complete the following conditional tasks in PDC/BRM and Launch.

   a. The Time Model and the respective time periods configured using Custom Profile Specification in Launch and Time Model on PDC side should be same.

      i. In Launch, use the REST API to create Time Model (Time Model is created using Custom Profile Specification with profile type 'TIME_MODEL') -

      ```
      https://<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/
      productCatalogReferenceManagement/v1/customProfileSpecification.
      ```

      The Time Model name and the respective time period names should match with PDC.
      Example: Time Model with name TIME-MODEL-01 and the time periods PEAK and OFF_PEAK.

      ii. In PDC, configure Time Model using Pricing tab in PDC dashboard.

   b. The Service Specification Attributes configured using Service Specification in Launch and Service Class Attributes in BRM side should be same.

      i. In Launch, use the REST API to create Service Specification with the Service Class Attributes configured as Service Spec Characteristics -

      ```
      https://<HOST>//crmRestApi/atcProductCatalog/11.13.18.05/tmf-api/
      serviceCatalogManagement/v3/serviceSpecification
      ```

      . The Service Class attribute names should match with PDC.
      Example: Service Specification with Service Spec Characteristic 'STATUS_FLAG'.

      ii. In PDC, ensure the Service Class attributes are configured in BRM.

      iii. Ensure the service class attributes (Service attributes used for Charge Selectors) configured in PDC matches the attributes added in Launch through Service Specification. These attributes will be used for Attribute Based Pricing.

   c. The Customer Profile Specification Attributes configured using Customer Profile Specification in Launch and Customer Attributes in BRM side should be the same.

      i. In Launch, use the REST API to create Customer Profile Specification with the Customer Attributes configured as Customer Profile Spec Characteristics-

      ```
      https://<HOST>//crmRestApi/atcProductCatalog/11.13.18.05/
      productCatalogReferenceManagement/v1/customerProfileSpecification.
      ```

      The Customer attribute names should match with PDC.

Example: Customer Profile Specification with Customer Profile Specification Characteristics 'NAME' and 'CURRENCY'.

ii. In PDC, ensure the Customer attributes are configured in BRM.

iii. Ensure the Customer attributes (used for Charge Selectors) configured in PDC matches the attributes added in Launch via Customer Profile Specification. These attributes will be used for Attribute Based Pricing.

d. Usage Specification name, metering rule name, and Usage Event Attribute names in Launch needs to be the same as the configured Usage Event Name, RUM name, and the Usage Event attribute names in PDC side.

i. In Launch, use the REST API to create the usage specification.

```
https://<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/tmf-api/
usageManagement/v2/usageSpecification. The fields name,
meteringRule.name should match with PDC.
```

The fields name, **meteringRule.name** and **usageSpecCharacteristic.name** should match with PDC.

Example: **EventDelayedSessionTelcoGsm, meteringRule.name** – Duration, **usageSpecCharacteristic.name** - CALLED_TO

ii. In PDC, configure the **serviceEventMap** with the same event name, RUM and Usage Event Attributes.

iii. Ensure the Usage Event attributes (Used in Charge Selectors) configured in PDC matches the attributes added in Launch via Usage Specification. These attributes will be used for Attribute Based Pricing.

e. The Service specification code in service specification in Launch and Service Event Map Name on BRM side should be same.

i. Launch Service Specification service code needs to be configured.

Example: The service code /service/telco/gsm/telephony needs to be same on both systems.

ii. In PDC, configure the ServiceEventMap.

f. The **Custom Analyzer Rule** configured using **Custom Profile Specification** in Launch and **Custom Analyzer Rule** on the BRM side should be same.

i. In Launch, use the REST API to create **Custom Analyzer Rule** (**Custom Analyzer Rule** is created using **Custom Profile Specification** with profile type 'CUSTOM_ANALYZER_RULE').

```
https://<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/
productCatalogReferenceManagement/v1/customProfileSpecification
```

The **Custom Analyzer Rule** name should match with PDC.
Example: Name: FAMILY_RULE

ii. In PDC, configure **Custom Analyzer Rules** using **Setup** tab in PDC dashboard.

g. The **Price Tag** in Launch and **Price Tag** on the BRM side should be same.

i. In Launch, use the REST API to create **Price Tags**.

```
https://<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/
productCatalogReferenceManagement/v1/priceTag
```

The **Price Tag** name should match with PDC.
Example: Name: PRICETAG_01

ii. In PDC, configure **Price Tags** using **Setup** tab in PDC dashboard.

h. The **Impact Category** in Launch and **Impact Category** on the BRM side should be same.

i. In Launch, use the **Business Configurations** section in **Administration** tab to configure the **Impact Categories**. The **Impact Category** name should match with PDC.
Example: Name: IC_INTERNATIONAL

ii. In PDC, configure **Impact Categories** using **Setup** tab in PDC dashboard.

i. The **Standard Zone** configured using **Custom Profile Specification** in Launch and **Standard Zone** on the BRM side should be same.

i. In Launch, use the REST API to create **Standard Zone** (**Standard Zone** is created using **Custom Profile Specification** with profile type 'STANDARD_ZONE')

```
https://<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/
productCatalogReferenceManagement/v1/customProfileSpecification
```

The **Standard Zone** name should match with PDC.
Example: Name: EUROPE_ZONE

ii. In PDC, configure **Standard Zone** using **Setup** tab in PDC dashboard.

j. The **Value Maps** configured using **Custom Profile Specification** in Launch and **Value Maps** on BRM side should be same.

i. In Launch, use the REST API to create **Value Map** (**Value Map** is created using **Custom Profile Specification** with profile type 'ZONE_VALUE_MAP')

```
https://<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/
productCatalogReferenceManagement/v1/customProfileSpecification
```

The **Value Map** name should match with PDC.
Example: Name: US_VALUE_MAP

ii. In PDC, configure **Value Map** in PDC instance.

k. The Policy Specification Terms configured using Custom Profile Specification in Launch and Policy Specifications on BRM side should be same.

i. In Launch, use the REST API to create Policy Specification Term(Policy Specification Term is created using Custom Profile Specification with profile type 'POLICY_SPEC_TERM') -

```
https://<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/
productCatalogReferenceManagement/v1/customProfileSpecification
```

The policy specification name should match with PDC. Example: Name: PCRF-1.

ii. In PDC, configure Policy Specifications using Setup tab in PDC dashboard.

Sample Mapping

Table 2-4 shows a sample mapping between Launch and PDC entities.

Table 2-4 Sample Mapping

Entity	Launch	PDC	Mandatory
Time Model	Create Custom Profile Specification with profile Type = 'TIME_MODEL'. Use it along with Product Offering. Time Model Name : TIME-MODEL-01 Time Periods : PEAK, OFF_PEAK	Time Model Name : TIME-MODEL-01 Time Periods : PEAK, OFF_PEAK	No
Service Specification	Create Service Specification with Service Specification Characteristics. Use it in Attribute Based pricing. serviceSpecCharacteristic.name : STATUS_FLAG	Service Attribute : STATUS_FLAG. This is used along with Charge Selectors	No
Customer Profile Specification	Create Customer Profile Specification with Customer Profile Specification Characteristics. Use it in Attribute Based pricing. custProfSpecCharacteristic.name : NAME	Customer Attributes: NAME This is used along with Charge Selectors	No
Usage Specification	EventDelayedSessionTelcoGsm Metering Rule: Duration Usage Spec Characteristic: CALLED_TO Use Usage Spec Characteristics along with Attribute Based Pricing.	EventDelayedSessionTelcoGsm (Event) RUM: Duration Usage Event Attribute: CALLED_TO	Yes
UOM (Unit of Measure)	ORA_ATC_UOM	UOM	Yes
Tax Code	TAX001	TAX001	No
Product Offering Billing Service Type	Populate the Service code - /service/telco/gsm/telephony in ORA_ATC_BILLING_SERVICE_TYPE and then use it on product offering billing service type	Service - /service/telco/gsm/telephony (BRM) serviceTelcoGSMTelephony(PDC) Configure the service-event maps.	Yes

Table 2-4 (Cont.) Sample Mapping

Entity	Launch	PDC	Mandatory
Product Specification	Wireless PS • Populate the Usage Specification - EventDelayedTelcoGSMTelephony Populate the Service Specification - Wireless CFS	N/A	No
GLID	101	101	No
Balance Element	Name: US Dollars Code: USD Numeric code: 840	Name: US Dollars Code: USD Numeric code: 840	Yes
Price Tag	CT01	CT01	No
Impact Category	Common business configuration impact category IC_INTERNATIONAL	IC_INTERNATIONAL	No
Custom Analyzer Rule	Create Custom Profile Specification with profile Type = 'CUSTOM_ANALYZER_RULE'. Use it along with product offering. Name: FAMILY_RULE	Name: FAMILY_RULE	No
Standard Zone	Create Custom Profile Specification with profile Type = 'STANDARD_ZONE'. Use it along with product offering. Name: EUROPE_ZONE	Name:EUROPE_ZONE	No
Value Map	Create Custom Profile Specification with profile Type = 'ZONE_VALUE_MAP'. Use it along with product offering.Name: US_VALUE_MAP	Name:US_VALUE_MAP	No
Policy Specification	Create Custom Profile Specification with profile Type = 'POLICY_SPEC_TERM'. Use it along with Product Offering. Name: PCRF-1	PCRF-1	No

Supported Scenarios

Table 2-5 lists the supported integration scenarios.

Table 2-5 Supported Scenarios

What you can publish?	Launch Entity	PDC Entity
Simple Offer with one time price	Simple product offering of service type Supported fee types are Purchase and Cancel	Charge offering of service type (EventBillingProductFeePurchase EventBillingProductFeeCancel)
Simple Offer with recurring price	Simple product offering of service type Supported recurring frequency - Monthly, Bi-Monthly, Semi Annual, Annual, Quarterly, Arrear and Forward Arrear	Charge offering of service type with the event of the below recurring frequency with the scaled fee. (EventBillingProductFeeCycleCycle_forward_annual - Occurrence EventBillingProductFeeCycleCycle_forward_semiannual - Occurrence EventBillingProductFeeCycleCycle_forward_quarterly - Occurrence EventBillingProductFeeCycleCycle_forward_bimonthly - Occurrence EventBillingProductFeeCycleCycle_forward_monthly - Occurrence) EventBillingProductFeeCycleCycle_arrear - Occurrence EventBillingProductFeeCycleCycle_forward_arrear - Occurrence)
Simple Offer with one time, recurring and usage fee (any metering rule)	Simple product offering of service type	Charge offering of service type with the one time, recurring, and usage fee
Simple Offer with a one- time, recurring price along with one time, recurring fixed/% discount	Simple offering with fees and adjustments of type fixed discount or percentage discount	Charge offering of service type with the one time, recurring, and usage fee. Discount offering with a fixed or percentage discount for the one-time fee
Simple Offer with a usage fee and usage discount	Simple offering with usage fees, metering rule, UOM with usage percentage or fixed discount	Charge offering of service type with the usage fee. Discount offering with a fixed or percentage discount for the usage fee
Simple Offer - Tiered pricing	Simple offer with one time/ recurring tiered pricing	Charge offer of service type with one time/recurring tiered pricing
Simple offer - Usage tiered pricing	Simple offer with usage tiered pricing	Charge offer with usage tiered pricing
Re-use of price plans in Simple offer	Simple offer with reused price plans	Charge offer with reused rate plans
Simple offer with Time Limited Discounts sync from Launch to BRM (Only absolute validity)	Simple offer with one time/ recurring/usage limited time discount	Charge offer/Discount offer with rate plan validity

Table 2-5 (Cont.) Supported Scenarios

What you can publish?	Launch Entity	PDC Entity
Launch - PDC - Scenarios involving revision, cloning, and retirement.	Clone Simple offer revision Simple offer Retire Simple offer Obsolete	Charge offer revise Charge offer obsolete Charge offer obsolete
Launch - PDC - Simple offering with allowances	Simple offering with single allowance	Charge offering of service type with the non- currency resource granted part of one time, recurring, and usage fee for consumption. Discount offering for non-currency resource consumption
Launch - PDC - Simple offering with Allowance and Overage	Simple offering with single allowance and overage	Charge offering of service type with the non- currency resource granted part of one time, recurring, and usage fee for consumption. Discount offering for non-currency resource consumption
Launch - PDC - Simple offering with Attribute based pricing (one time, recurring and usage)	Simple offer with attribute-based pricing for usage with service specification and usage specification characteristics. Simple offer with attribute-based pricing for one time and recurring with service specification and customer specification characteristics.	Charge offer with charge selector
Launch PDC - Simple offering with one-time, recurring fees and discounts along with Charging terms	Simple offer with charging terms	Charge offer/Discount offer with rate plan configuration or charging details like proration and increments
Launch - PDC - Simple offering with one-time, recurring fees and usage fees (reuse of price plan alteration)	Simple offer with reused discount price plan	Discount offer with reused discount rate plan
Simple Offer with a one time, recurring, usage volume discount (tiered, volume)	Simple offer with one time/ recurring/usage tiered and volume discount	Charge offer and Discount offer with one time/recurring/usage tiered and volume discount
Service Bundle (No nesting of bundles)	Service bundle	Bundle
Package with commitment terms (no nested bundles or commercial bundles, aggregate groups)	Package	Package with commitment terms
Service Bundle with commitment terms (No nesting of bundles)	Service bundle	Bundle with commitment terms
Package (no nested bundles or commercial bundles, aggregate groups)	Package	Package

Table 2-5 (Cont.) Supported Scenarios

What you can publish?	Launch Entity	PDC Entity
Simple Offer with attribute-based adjustments for usage	Simple offer with attribute-based adjustment for usage (customer specification, service specification, usage specification)	Charge offer and Discount offer with Discount Selector
Publish other type of product offering (like discounts)	Simple offering with other type of product offering + discount	Discount offers
Simple Offer with multiple allowances and consumption model	Simple offer With multiple allowances and consumption model	Charge offer and Discount offer (consumption model)
Simple Offer with usage prices based on zoning	Simple offer with Value Map zoning	Charge offer using charge selector with value map zoning
Simple Offers with adjustments based on triggers	Simple offer, triggers on adjustments (Total Charge, Total Quantity, Price Tag, Expression, Price Plan Name, Currency as trigger conditions)	Discount offer with Discount Trigger (Charge Selector Spec and Trigger Spec are used to configure these properties)
Simple offers with multiple price lists	Simple offers with multiple price lists. (Fees are created with different price lists within the same simple offers.)	Charge offers with Charge selectors. The charge selectors are used to configure charge rate plans based on price lists.
Simple offers with adjustments and user/share balance	Simple offers with adjustments and user/sharer balance	Discount offers with different types of discounts which applies to user/sharer balance.
Simple offers with Balance Consumption Model	Simple offers with Balance Consumption Model without Allowance configured in usage fee (Consumption Discount Model).	Discount offers with Balance Consumption Model
Simple offers with standard zone	Simple offer with attribute based pricing for usage based on standard zone.	Charge offer using charge rate plan with standard zone
Simple offers with price tags for run time price overrides	Simple offer with price tags on product offering prices.	Charge and discount offers with rates having price tags
Simple offers with multiple usage prices	Simple offers with multiple usage prices. Attach appropriate usage specification for each usage fee.	Charge and discount offers with multiple usage events
Simple offers with Allowance Grant	Simple offers with provision to grant allowance as adjustments. This can be done either as a new adjustment or along with an existing adjustment where the product type is DISCOUNT.	Alteration Rate Plans with Allowance Grant
Simple Offers with Custom Analyzer Rule for Pricing	Simple Offers with provision to configure pricing based on Custom Analyzer Rule. This is done by adding Custom Analyzer Rules along with Attribute-based Pricing.	Charge Offers with Charge Rate Plan along with Generic Selector configured inside the Charge Rate Plan

Table 2-5 (Cont.) Supported Scenarios

What you can publish?	Launch Entity	PDC Entity
Simple Offers with Discount Priority	Simple Offers of product type Discount/Time Based Discount with discount priority.	Discount Offers with Alteration Rate Plan
Simple Offers with Adjustments and Discount Mode - Parallel/ Sequential	Simple Offers with Adjustments and Discount Mode - Parallel (Original Charge)/Sequential (Remaining Charge)	Discount Offers with Alteration Rate Plan
Simple Offers with Counters	Simple Product Offerings with provision to configure Counters with Pricing, Adjustments.	Charge Offers with Charge Rate Plan and Discount Offers with Alteration Rate Plans
Simple Offers with Start Validity Mode for Allowances and Counters	Simple Product Offerings with provision to configure start validity mode for Allowances (including allowance grants) and counters.	Charge Offers with Charge Rate Plan and Discount Offers with Alteration Rate Plans
Simple Offers with Policy Specification	Simple Product Offerings with provision to configure Policy Specification.	Charge Offers and Discount Offers
Simple Offers with Roll Over Configurations	Simple Product Offerings with provision to configure Rollover Configurations.	Charge Offers with Alteration Rate Plan
Simple Offers with Charge Share	Simple Product Offerings with provision to add Charge Share	Charge Share Offers (Distribution Offerings with Distribution Rate Plan)
Simple Offer with Time Model	Simple Product Offering with Time Model tagged to the price plans	Charge Offering with Time Model tagged to the Charge Rate Plans
Simple Offer with Formula Based Discount	Simple Product Offering with Formula Based Discounts	Discounts Offers with expressions added to the Alteration Rate Plans

3

Integration with Service Catalog and Design (SCD)

Telecommunications catalogs fall into two broad categories: **commercial catalogs** and **technical catalogs**.

The **commercial catalog** is customer-facing. Launch is the commercial catalog application. It defines what you sell and how you present it across engagement channels. It manages commercial products and services, promotions, pricing, rules, and terms.

The **technical catalog** is internal-facing. Oracle Communications Service Catalog and Design (SCD) is the application that manages the technical catalog. It defines the services and resources (physical and logical) required to fulfill and operate the commercial offerings.

Together, Launch and SCD provide an end-to-end catalog foundation that helps you bring compelling offers to market. The prebuilt integration between Launch and SCD enables centralized, design-time modeling of both commercial and technical catalog portfolios across your ecosystem.

Related Guides

Table 3-1 contains information about other useful sources of information for the integration process.

Table 3-1 Related Guides

References	Description
Launch Cloud Service User's Guide	Describes how you can create, publish, and manage product offers.
REST API Reference for Launch Cloud Service	Provides the REST API reference document for Launch Cloud Service.
Oracle Digital Experience for Communications Industry Fabric Implementation Guide	Describes the setup and implementation of the CX Industries Framework required to deploy Launch Cloud Service.
Service Catalog and Design External Product Catalog Integration Guide	Describes how to integrate Oracle Communications Service Catalog and Design Solution Designer with an external product catalog such as Launch.

Supported Versions

- Launch version 26.04 or later
- SCD version 8.3.0.1 or later

Supported Integration and Mapping

Table 3-2 shows all the entities that are released from SCD and published from Launch.

Table 3-2 Supported Integration and Mapping

Launch Entity	SCD Entity	What can you sync?
Product Specification	Product Specification	• Definition • Attributes or Data Elements • Attribute or Data Element Definition • Attribute or Data Element Values • Product Specification (PS) to Customer Facing Service (CFS) characteristic Mapping
Service Specification	Customer Facing Service (CFS)	• Definition • Attributes or Data Elements • Attribute or Data Element Definition • Attribute or Data Element Values

Setting Up Launch SCD Integration

Table 3-3 lists the setup tasks you need to perform in Launch, Industry Framework, and SCD for bringing CFS and its design parameters from SCD to Launch and subsequent publishing from Launch to SCD.

Table 3-3 Setup Task List

No.	Application	Task	Mandatory?	Description
1.	Industry Framework	Create Integration User	Yes	This is required to facilitate the integration between the two applications.
2.	Industry Framework	Configure the spoke systems	Yes	This is required to ensure to configure the spoke system instance for receiving publishing events.
3.	Industry Framework	Patch GKR to configure Json Schema's	Yes	This is required to claim the resources and configure JSON schemas to validate the incoming data.
4.	Launch	Register destinations	Yes	This is required to configure the right spoke instance to publish the launch or projectPublishEvent to SCD.

Table 3-3 (Cont.) Setup Task List

No.	Application	Task	Mandatory?	Description
5.	Launch	Configure Feature Profile	Yes	This is required to restrict CFS creation in Launch UI when SCD is part of solution
6.	SCD	Create client credentials and security requirements	Yes	N/A

Ownership and Consumption Pattern

Table 3-4 outlines the responsibilities of Launch and SCD regarding the consumption and publishing of key entities in the integration process.

Table 3-4 Ownership and Consumption Responsibilities

Launch Entity	Ownership	Consumer
Product Spec to CFS Spec mapping and its characteristics	Launch	SCD
Product Specification	Launch	SCD
Service Specification	SCD	Launch

Setup Task Details

To set up SCD and Launch for catalog migration and publishing, follow these steps:

1. Create a new user on the security console with the username **FABRIC_SYSTEM_USER** and assign the role **Communications Catalog Administrator**. If this user was already created during the Launch setup, no further action is required.
2. Configure your external application in the Industry Framework for each instance. For more information, refer to the topic Integrate External Applications to add a Spoke End Point in the article Implement CX Industries Framework, on My Oracle Support, Doc ID 2720527.1.

> **Note**
>
> When creating the `ConnectionDescriptor`, ensure to use the DNS-mapped address instead of the raw IP address (for example, `100.111.86.234`) for setting `<oidc-client-credentials> and <endpoint-url>` in the `ConnectionDescriptor`.

3. Once the connection descriptor (TIC) is created, an empty GatekeepingRule is automatically generated with an ID in the format `gkr-<connectionDescriptorId>`. For example, if the connection descriptor ID is scdl59zt, the GatekeepingRule ID will be gkr-scdl59zt.

   We need to update this empty gkr to claim the resources and configuring Json Schemas to validate eventing payload coming from SCD app. Please follow below steps to update the gkr-

Retrieve the GatekeepingRule (GKR), make a GET request passing `gkr-<connectionDescriptorId>`.

Endpoint: {{FA_APIGW}}/admin/gatekeepingRules/{gkr-<connectionDescriptorId>}

Method Type: GET

Sample GKR - GET Call Response

```
{
  "endpoint-name": "<auto-generated using endpoint-name from TIC>",
  "rule-name": "Generated gatekeeping rule for endpoint scd",
  "id": "<generated gkr-id>"
}
```

> **Note**
>
> Ensure to replace the gkr id correctly in above endpoint request. **FA_APIGW** in URI is Fusion Application Gateway.

Next step is to update the **GatekeepingRule** response retrieved in above get call response by manually adding the "destination-selection" section from below sample request body. Once the request body to update gkr is prepared, please use below endpoint to update the gkr.

Endpoint: {{FA_APIGW}}/admin/gatekeepingRules/{gkr-<connectionDescriptorId>}

Method Type: PUT

Sample Request Body

```
{
    "endpoint-name": "<auto generated using endpoint-name from TIC>",
    "rule-name": "Generated gatekeeping rule for endpoint scd",
    "destination-selection": [
        {
            "api-id": "tmf-633",
            "api-version": "v4",
            "criteria": [
                {
                    "rank": 10,
                    "resource-ids": [
                        "serviceSpecification"
                    ],
                    "event-schemas": {
                        "ServiceSpecificationReleaseEvent": {
                            "json-schema-ref": "service-specification-
release-event.schema.json"
                        },
                        "ProjectStateChangeEvent": {
                            "json-schema-ref": "project-state-change-
event.schema.json"
                        }
                    }
                }
            ]
```

```
        }
    ],
    "id": "<generated gkr-id>"
}
```

> **Note**
>
> Ensure to replace the gkr id correctly in above endpoint request. **FA_APIGW** in URI is Fusion Application Gateway.

4. Register the destination in Launch for publishing to the SCD instance. You must set up the SCD instance in Launch to ensure that the publishing is directed correctly.

   a. Configure destinations in Launch to the appropriate lifecycle status for which you want to publish to the external application. For example, configure the test instance destination to the Ready to Publish lifecycle status and the production instance destination to the In Design status.

   b. While creating SCD destination, use the following settings:
   Name: The target pre-selection key from the Connection Descriptor in the Industry Framework

   Type: The target name used in the Catalog Sync configuration.

   Internal: Set to **OFF**.

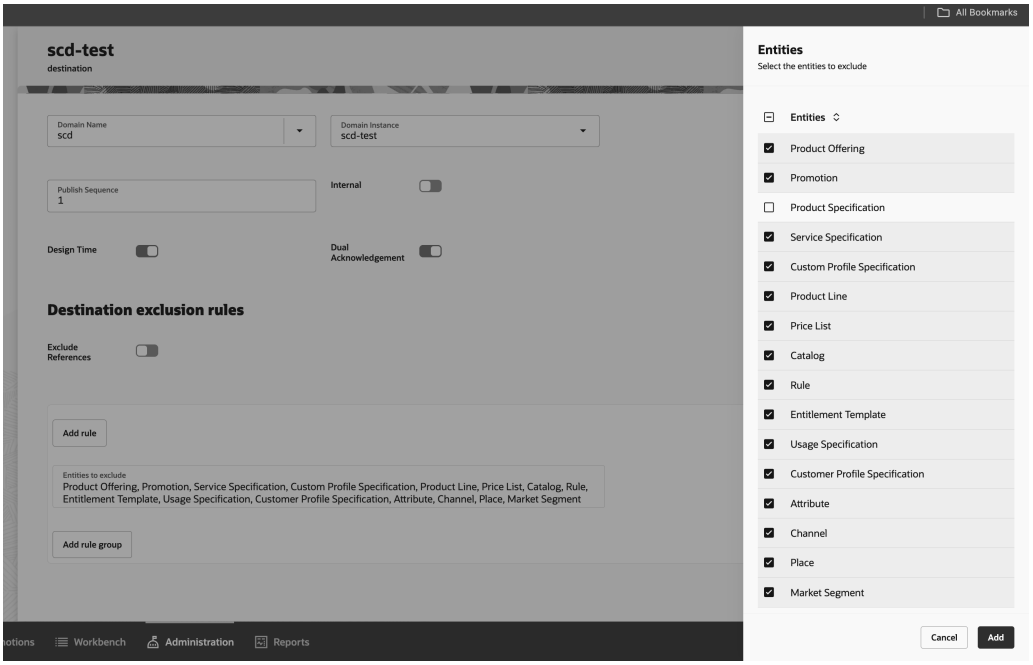
   Exclude References: Set to **OFF**.

   Design Time: Set to **ON**

   Dual Acknowledgement: Set to **ON**

   > **Note**
   >
   > - A Design Time flag is used to differentiate design-time destinations from runtime destinations within the Launch app.
   > - Dual Acknowledgment indicates that Launch expects two acknowledgments from SCD for its /projectPublishEvent
   >
   >   i. **First acknowledgment**: When SCD consumes the **/projectPublishEvent**
   >
   >   ii. **Second or Final acknowledgment**: When Launch provided initiative in SCD moves to Functional testing indicating Launch that SCD is ready for publishing to its RunTime app.

   c. Create a rule to exclude all entities except Product Specification (i.e select all entities except Product Specification). Please refer below snapshot.



d. To configure a lifecycle status for SCD destination,

   i. Navigate to the Launch user interface, click the **Administration** tab, then click **Lifecycle Status**.

   ii. Select the new lifecycle configuration version in the PENDING state.

   iii. Choose **Edit** for the Ready To Publish status and click **Add Destination**.

   iv. Provide the Name, Type, and Publish Sequence. Click **Add**.

> **Note**
>
> The SCD sequence value should be set to a value before any other runtime application sequences, as it is a design-time app. Both Launch and SCD configurations should co-develop the technical configuration first. Subsequent sequences for PDC, Siebel, and other runtime systems should be set accordingly. When assigning publishing sequences, it is recommended to leave intentional gaps between them to allow easy addition in the future without disrupting the existing sequence.

   v. Select the SCD destination name in the Destinations field and save.

   vi. Choose **Activate** for the PENDING lifecycle configuration version.

   vii. Verify that the lifecycle configuration version status changes to **ACTIVE**.

   For more information on how to configure the Destination in Launch, see *REST API Reference for Launch Cloud Service* and Publish Catalog Entities in *Launch Cloud Service Implementation Guide*.

5. Make feature flag inactive to restrict CFS creation in Launch UI when SCD is part of the solution.
   Method Type: POST

   Endpoint: {{FA_APIGW}}/api/productCatalogManagement/v1/featureProfile

Request Body

```
{
    "CFS_CREATION": "Inactive"
}
```

> **Note**
>
> Once the feature flag is set to **Inactive**, it cannot be changed back to **Active**. **FA_APIGW** in URI is Fusion Application Gateway.

6. Create client credentials and define the security requirements for Oracle Communications Service Catalog and Design (SCD). See the *REST API for Oracle Communications Service Catalog and Design* for detailed instructions and configuration steps.

Supported Scenarios

Table 3-5 lists the supported integration scenarios.

Table 3-5 Supported Scenarios

S.No	Emitting App	Receiving App	Use Case	Description
1.	SCD	Launch	Release an initiative with Customer Facing Service (CFS) to Launch	Release an initiative with CFS and its associated attributes by changing the initiative lifecycle status from "Definition" to "Released" state in SCD.
2.	Launch	SCD	Publish an Initiative with Product Specification (PS)	Publish an initiative with Product Specification (PS), associated attributes, and Product Specification (PS) to Customer Facing Service (CFS) mapping to SCD.
3.	SCD	Launch	Revision of Customer Facing Service (CFS)	Revise the CFS in the new initiative by making changes to associated attributes or to the definition, and release the new initiative to Launch from SCD.

Table 3-5 (Cont.) Supported Scenarios

S.No	Emitting App	Receiving App	Use Case	Description
4.	Launch	SCD	Revision of Product Specification (PS)	Revise the PS in the new initiative by making changes to the associated attributes, the definition, or the PS-CFS characteristic mapping, and publish the new initiative to SCD.

Launch-SCD Integration Events

Figure 3-1 shows the Launch-SCD integration events. The SCD admin releases an initiative with the Customer Facing Service (CFS) using the **serviceSpecificationReleaseEvent**. After which, Launch sends an acknowledgment (either FAILURE or SUCCESS) to confirm the event has been received.

Figure 3-1 serviceSpecificationReleaseEvent

The Launch admin publishes the Product Specification (PS) using the projectPublishEvent. Then, the SCD admin sends an initial acknowledgment, publishingAcknowledgement, confirming that the event was received. Later, the SCD admin sends a

projectStateChangeEvent, indicating that the event has been consumed and is ready to be published to the runtime systems.

Figure 3-2 projectStateChangeEvent

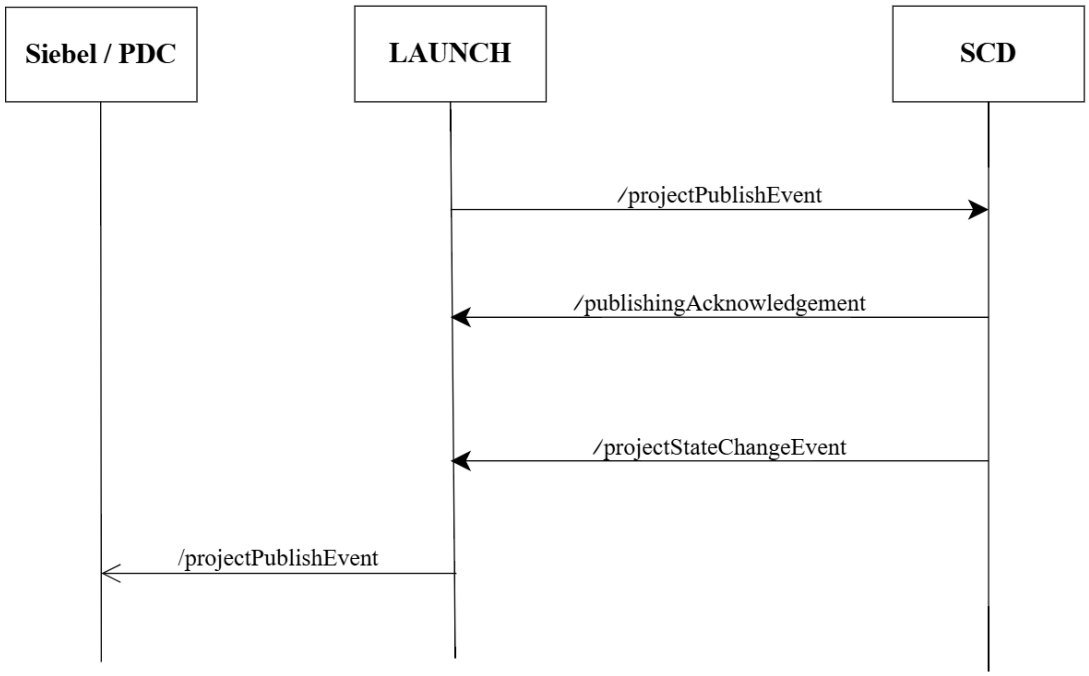


Table 3-6 describes the list of events that are triggered for the flow of data between Launch and SCD.

Table 3-6 Supported Events for Launch-SCD Integration

S.No	Emitting App	Receiving App	Event Endpoint	Description
1.	SCD	Launch	/ serviceSpecificatio nReleaseEvent	The SCD Admin creates a new initiative, adds Customer Facing Service (CFS) with its associated attributes, and releases the initiative using the / serviceSpecificatio nReleaseEvent event.

Table 3-6 (Cont.) Supported Events for Launch-SCD Integration

S.No	Emitting App	Receiving App	Event Endpoint	Description
2.	Launch	SCD	/eventAcknowledgement	The Launch app, after receiving the /serviceSpecificationReleaseEvent, sends an event acknowledgment to SCD indicating whether the event released was consumed or not, by sending a success or failure acknowledgment using the /eventAcknowledgement event.
3.	Launch	SCD	/projectPublishEvent	The Launch Admin creates a new initiative, creates Product Specification (PS) with associated attributes, performs Product Specification (PS) to Customer Facing Service (CFS) characteristic mapping, and publishes it to SCD using the /projectPublishEvent event. The CFS released from SCD is to be associated with the Product Specification (PS).
4.	SCD	Launch	/publishAcknowledgement	The SCD app, after receiving the /projectPublishEvent, sends an acknowledgment to Launch indicating whether the event released was consumed or not, by sending a success or failure acknowledgment using the /publishAcknowledgement event.

Table 3-6 (Cont.) Supported Events for Launch-SCD Integration

S.No	Emitting App	Receiving App	Event Endpoint	Description
5.	SCD	Launch	/projectStateChangeEvent	After the Launch app consumes the /publishAcknowledgement event from SCD, the SCD app sends the /projectStateChangeEvent event to indicate its readiness to publish to respective runtime apps.
6.	Launch	Run Time Apps	/projectPublishEvent	The Launch Catalog Admin publishes Initiative to the rest of its runtime destinations (Siebel and PDC) using the /projectPublishEvent event.

Troubleshooting Tips

Table 3-7 lists the troubleshooting tips for integration with SCD..

Table 3-7 Troubleshooting Tips

S.No	Issue	Reason	Resolution Tip
1.	An attribute with the matching ID {attribute_id} already exists in Launch.	The attribute with this specific ID is either already created in Launch or migrated from its runtime apps. An attribute can only be created if it was originally brought from SCD, or if no attribute already exists with the same ID in Launch.	Ensure that the attribute was originally brought from SCD or that no other attribute with the same ID exists in Launch.

Table 3-7 (Cont.) Troubleshooting Tips

S.No	Issue	Reason	Resolution Tip
2.	Attributes are not created in Launch even if the event is successful.	There are attribute data types not supported in Launch, such as ['HEXBINARY', 'FEATUREGROUP', 'TIME']. Therefore, Launch ignores these and only creates attributes with supported data types. Download the error log file to view the validation message, which will indicate which CFS (CFS_name) have unsupported data type attributes.	Check the data types of the attributes. If unsupported types like ['HEXBINARY', 'FEATUREGROUP', 'TIME'] are present, modify them. Download the error log file for more details.
3.	The [attribute_value] values for the [attribute_name] attribute can't be deleted once they've been created.	Attribute values cannot be deleted during revision or update in Launch once they have been created and used across other entities.	Ensure the attribute value is not deleted during the revision or update of attributes once they are created in Launch.

4

Integration with Third Party Content Management System

Integration between Launch and a Headless third party Content Management System (CMS) is required to manage content across your ecosystem.

Introduction

This comprehensive guide provides step-by-step instructions for implementing a third-party Headless Content Management System (CMS) integration with Launch. The integration allows for efficient content management and retrieval, enhancing the overall functionality of the Launch platform. The document is compiled with the assumption that there is a concrete implementation of the third-party CMS Swagger provided by Launch.

Purpose

The purpose of this integration is to enable Launch to interact with external CMS systems, providing flexibility in content management and allowing for seamless content retrieval and display within Launch.

Scope

This chapter covers the entire process from initial configuration in Fabric to final testing in Launch, including API setup, authentication configuration, and Launch-specific settings.

Prerequisites

Before beginning the implementation, ensure you have the following:

- Access to the CXIFHost environment (e.g., https://your-cxif-host-example.com)
- Access to the FAHost environment (e.g., https://fa-host-example.com)
- Tenant Admin privileges for running /admin APIs
- Visual Builder Studio access with appropriate permissions
- Authentication credentials for the third-party CMS (OAuth2, Basic Auth, or OCIHttpSignature)
- Familiarity with RESTful APIs and JSON
- Access to curl or Postman for API testing
- Understanding of CORS (Cross-Origin Resource Sharing) concepts
- Knowledge of the specific third-party CMS being integrated

Related Guides

Table 4-1 contains information about other useful sources of information for the integration process.

Table 4-1 Related Guides for Third Party Content Management System Integration

Reference	Description
Implement CX Industries Framework	Describes the setup and implementation of the CX Industries Framework required to deploy Launch Cloud Service.

Supported Versions

- Launch version 24.10 or later

System Architecture Overview

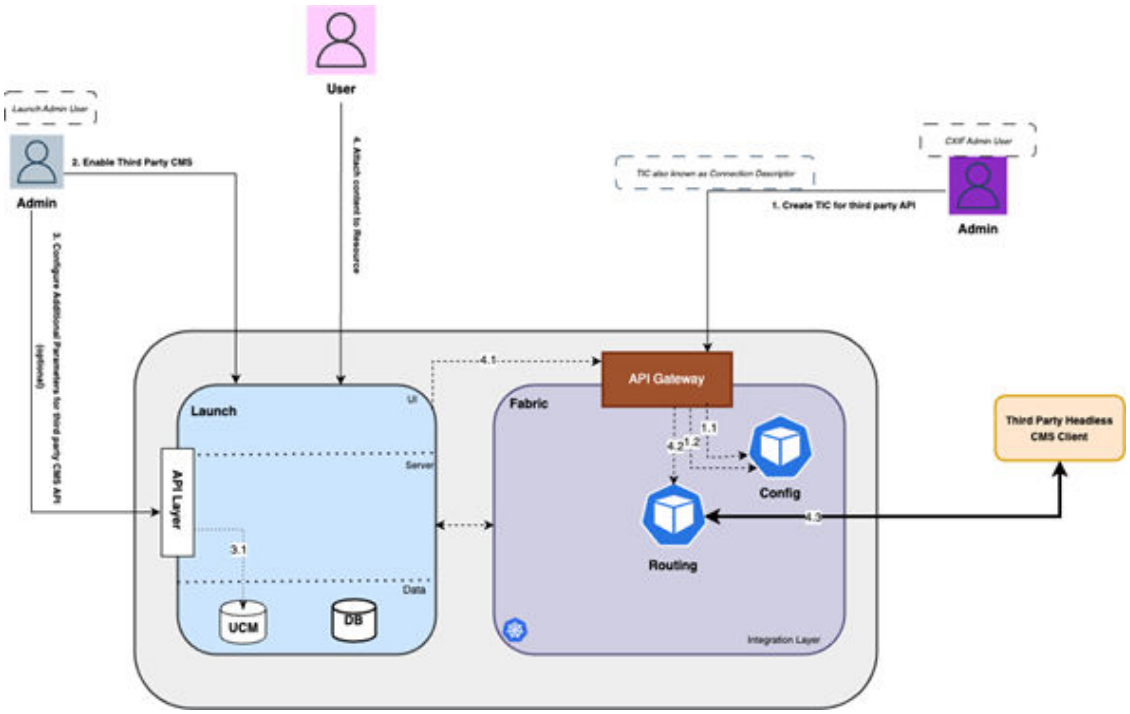
The integration involves three main components:

1. **Fabric**: Acts as the API management and routing layer.
2. **Launch**: The application platform that will consume CMS content.
3. **Third-party CMS Client**: The external content management system.

The flow of data is as follows:

Launch > Fabric > Third party CMS

Figure 4-1 System Architecture Overview



The diagram illustrates the architecture for integrating and the flow of third-party Content Management System (CMS) functionality.

1. Assuming the CMS Client is ready, the process begins with creating a TIC or Connection Descriptor and updating the Gatekeeping Rules using the Admin API. This TIC specifies authentication method, credentials, and host information of the CMS client.
2. Launch must be configured to recognize the third-party CMS, this configuration is done using Visual Builder Studio.
3. If the CMS client needs additional parameters with each request, these are set up as a JSON configuration in Launch Universal Content Management (UCM) at this path: *attachment/thirdPartyCMSParamters/AdditionalParams.json.*
4. Once configured, users can interact with the system. When editing a resource like Product Offering, clicking **Add Images** or **Add Documents** triggers a network call to the third-party CMS client using Fabric.

Finally, Launch renders the content in a drawer, allowing users to select and attach items to their resource. This setup enables seamless integration of external CMS capabilities within the existing system architecture.

Configuring Third Party Content Management System

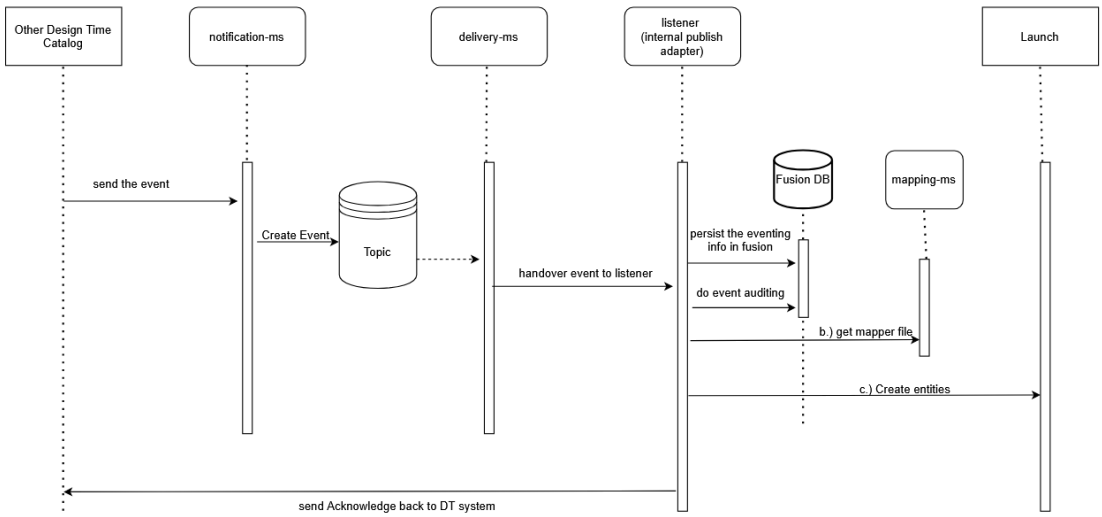
To configure the third party Content Management System:

1. Import the Postman collection (Collection: **Object Storage Spoke Integration.postman_collection.json**).
2. In the variables section, add all the variables that are in upper case.
3. In the auth section, authenticate.
4. In the run section, run the script with a 30ms delay.

5

Configuring Listener Endpoints for Third-Party Catalogs

Launch enables you to integrate external and internal catalogs such as partner, OTT, and other CSP catalogs by configuring a Listener flow. The Listener configuration follows an event-driven architecture where you create a listener endpoint in Launch, and the third-party catalog invokes that endpoint. When the endpoint is invoked with an appropriate payload, you can define data transformations and ingestion logic in the listener flow's mapping file, similar to Migration and Publish flows. The diagram below illustrates an overview of the Listener flow.



Set Up the Integration

Before creating the Listener mapping file, ensure that the spoke systems are registered and the integration between the applications has been validated. Once verified, proceed with the steps below:

1. Log in to Launch and navigate to Administration, select Catalog Sync, and then click Create.
2. Select Type as Listener and enter the Source name.
3. Upload the Source Grammar file. This file is required to understand the API capabilities and schema of the source environment.
4. (Optional) Upload the Target Grammar file if there are Launch-side extensions.
5. Click Save to complete the first checkpoint of the Listener mapping file creation.
6. Once saved, click Add Listener API to move to the next checkpoint.
7. Under General Information, enter a name for the Listener API. This is the endpoint that third-party catalogs will call to trigger the listener flow.

8. Depending on the event payload contract, choose one of the following approaches:
    a. If the payload contains only references to the entities that need to be synchronized, define API mappings to call the source system and retrieve the data.
    b. If the payload contains the full details of the entities to be synchronized, you do not need to call the source API. Proceed directly to the Target API invocation and data transformations.
9. Refer Productized Integration Mapping: Functions and Mapping Rules for details on configuring API Mapping and Data Mapping.
10. After the Listener API is defined, save this checkpoint to proceed to Target API definitions.
11. Add a Target API for each call to the target system. Provide the API details and specify data mappings/transformations in the data mapping table. Refer Productized Integration Mapping: Functions and Mapping Rules for more details.
12. Once saved, the mapping file is ready to be used by the listener flow.
13. When a third-party catalog triggers the event defined in step 7 with the contractual payload, the listener flow will read the mapping file and run the logic defined by the customer.

6

Productized Integration Mapping: Functions and Mapping Rules

Launch Integration uses a mapping service which enables you to create proxy API that can pull data into Launch and push data into external applications. The mapping service also allows customers to modify the seeded mapping rules using extensions. For more information, see "Siebel Extensions Handling".

API Mapping

The API mapping enables you to create a proxy API that can map APIs between the source and target systems.

The section in the API mapping are:

- Source API Mapping
- Target API Mapping

Source mapping allows you to orchestrate calling APIs on a source system and then map them to a set of target APIs. In case of publishing, the source is Launch and target is the external application and is vice versa in the case of migration.

Refer to respective Swagger Open API Specification (OAS) to understand more on the APIs. Refer to *Launch REST API Guide* to get the Launch Swagger.

Data Mapping

Data mapping enables you to create mappings between the target API schema and the logical source schema created using the source API Mapping. The mapping is done using JSON path expressions. The framework provides additional functions to simplify complex mapping that can't be supported using the JSON path.

You can use these built-in functions to map third-party catalog consuming apps using the supported functions and application constants.

Supported Functions

We may need to write java code to perform field mapping between Launch and other spoke system. Launch Integration provides below set of java functions which can be used in the mapper file.

Table 6-1 lists the functions available for use in the data transformation, refer to the description column to understand the use of a function.

Table 6-1 Supported Functions

Function Name	Signature	Description
@toUpper	@toUpper(json path)	Converts data to upper case.

Table 6-1 (Cont.) Supported Functions

Function Name	Signature	Description
@toDataType	@toDataType(value json path, data type json path, boolean true value, boolean false value)	Converts data to data type if the source data doesn't conform to the given data type or uses custom data type
@toFlatten	@toFlatten(source json path, relative depth json path)	Flattens an object or array to the given depth.
@indexOf	@indexOf(array json path)	Returns index of an array element.
@replaceValue	@replaceValue(jsonPath/ function/default constants, header key)	Uses header value passed in the request. The name of the parameters should be in this form - User-Project-Name. The first letter and the subsequent letters after hyphen(-) must be capitalized. Examples: @replaceValue($,User- Project-Id) # correct @replaceValue($,User-Project-Name) # correct @replaceValue($,user-project-name) # Incorrect
@invokeRest	@invokeRest(function name, argument list...)	Tenant implements end point and configures it in the Industry Framework.
@valueMap	@valueMap(condition json path/ function, default, json path/ function, source, target)	Maps the value found at the third argument using a variable number of source/target values. Keep in mind that when the target field uses an Open API type of "oneOf": [...], this function will not be able to cast mapped values to types unless the target type is declared with an explicit Open API "type" instead of "oneOf".
@conditionValueMap	@conditionValueMap (json path/ function, default, trueValue, falseValue)	Returns trueVal if the first argument resolves to anything non- null that is not an empty list. Returns falseValue if the first argument resolves to an empty list. Returns the default value if the first argument resolves to null.
@defaultValue	@defaultValue (json path/ function, default value)	Returns the value found after evaluating the first argument. Returns the default value if the first argument is null.

Table 6-1 (Cont.) Supported Functions

Function Name	Signature	Description
@distinctValues	@distinctValues(json path array, arg1, arg2, ...)	Returns the distinct values based on the parameter passed. The first argument should return an array. The other arguments contain fields that are used to find distinct elements.
@alterValue	@alterValue(condition json path, json path or value, mathematical operator, json path or value, precision(optional), rounding mode(optional))	It performs a mathematical operation on its operands and produces a number according to the specified precision and rounding mode.
@toArrayLength	@toArrayLength(condition json path, json path or function)	It calculates the length of an array and produces a number according to that.
@concatValues	@concatValues(condition, arg1, arg2, ...)	Returns the concatenated string based on the parameters passed. The 1st argument (function or JSON path) is a condition. If the first argument evaluates to not null, then the function will return the concatenated value else it will return null.
@compareValue	@compareValue(condition json path, json path or value or header param or query param, relational operator, json path or value, dataType optional)	It compares the left and right operands using the relational operators like (<,>,<=,>=,==,!=), based upon the dataType mentioned and returns a Boolean value as result.
@conditionalConvertValue	@conditionalConvertValue(conditionJson Path, jsonPath, regex, index, case identifier)	This function is used to split and convert the case of an input text passed in its 2nd arg to the user requested case passed in 3rd arg, 4th arg, and 5th arg.
@convertCase	@convertCase(condition json path/function, json path/function/ constants, case identifier)	This function is used to split and convert the case of an input text passed in its2nd arg to the user requested case passed in 3rd arg.
@dateConverter	@dateConverter(condition 'json path',date 'json path', default date(optional), source date format(optional), target date format(optional))	Based on the condition, it converts date from source format to target format if provided or uses default date format. If the date resolves to null and then returns default date converted in target format, if default date is also null then returns null.
@evaluateDataType	@evaluateDataType(jsonPath, dataType, format Optional)	It checks if the value of that jsonPath returns the specified data type or not. In case of date format has to be passed.

Table 6-1 (Cont.) Supported Functions

Function Name	Signature	Description
@filterArray	@filterArray(conditionalJsonPath, filteredArrayJsonPath, minLengthForComparing, operator, additionalCondition)	Filter the array based on the condition provided.
@getArrayElementByIndex	@getArrayElementByIndex(condition json path or function, json path or function, integer constant or json path or function returning integer index)	It returns an array element present at specified index passed in 3rd argument.
@getFilteredArrayElement	@getFilteredArrayElement(condition,jsonpath, filterCriteria, index, fieldsJsonpath)	Based on the condition, it substitutes the search string with replace string in the data and returns the transformed data.
@infixToPrefix	@infixToPrefix($, $.['stringExpression'], operator_map('REQUIRES','req','NOT','!'), type_map('REQUIRES','binary','!','unary'), operand_map(), exp_json_path($.id, productId), apply_Siebel_Format (true))	This function is used to convert infix expression to its prefix form.
@objectToStringExp	@objectToStringExp (jsonPath, keyValueSep, fieldSep, mode, [replaceString], [enclosedBy])	Function to convert a json object to a string, with the ability to ignore empty fields and separate fields, values, and field-value pairs, as well as wrap each field value pair with a string.
@prefixToInfix	@prefixToInfix(condition json path, json path, source_system_siebel (true/false), operator_map('source-system- operator','eqv- target-system- operator', ……),type_map('source- system-operator','operator- type', …….),operand _map('custom-source-system- operand','regex-exp-to-be-used or way to resolve custom operand in target system', …….),exp_json_path(entity(entity _Type_Identifier_In_Target_System, JsonPath to identify ID in stitched payload), ...))	This function is used to convert prefix expression to infix expression.
@splitString	@splitString(conditionJsonPath, jsonPath, regex, defaultValue, index)	It splits any string based on a regex and returns list of values if no index given in the argument else returns that specific value based on the index mentioned in the argument.

Table 6-1 (Cont.) Supported Functions

Function Name	Signature	Description
@substituteValues	@substituteValues(condition 'json path or function', data 'json path or value or function' , search string 'json path or value or function' , replace string 'json path or value or function', ...)	Based on the condition, it substitutes the search string with replace string in the data and returns the transformed data.

Supported Application Constants

Table 6-2 lists the application constants available for use in data transformation, refer to the description column to understand the use of respective application constants.

Table 6-2 Supported Application Constants

Name	Sample Payload	Description
@jobId	`"name": { "type": "string", "x-oracle-map-data": { "json_path": "@concatValues($,-,SM,@jobId)", "mapType": "TO_FIELD" } }`	The current job ID.
@applicationName	`"source":{ "type": "string", "x-oracle-map-data": { "default": "@applicationName" } },`	Returns the name of the source system.
@epochTime	`"startDate":{ "type":"string", "format":"date-time", "x-oracle-map-data":{ "default": "@epochTime" } }`	Returns the epoch time based on system time.

Table 6-2 (Cont.) Supported Application Constants

Name	Sample Payload	Description

@recordNum	<pre>"id":{
	"type":"string",
	"x-oracle-map-data":{
	"mapType":
	"TO_FIELD","concat":
	{
	"separator":"-",
	"fields":[
	"@const(Conditon)",
	"@recordNum"
	]
	}
	}
	}</pre>
		Returns the record number of the array element currently being processed.

@replace(key)	<pre>"source_request_spec": {
	"source_api_path": "/
	siebel/v1.0/data/Price
	List/Price List",
	"source_request_type":
	"GET",
	
	"distinct_key_json_path"
	: "$.Id",
	"select_json_path":
	"$.items",
	"query_param": [
	{
	"key": "PageSize",
	"value":
	"@replace(batchSize)"
	}
	]
	}</pre>
		Replaces a value from given header key.

Table 6-2 (Cont.) Supported Application Constants

Name	Sample Payload	Description

@pathParam(key)	<pre>{
	"source_api_path": "/
	siebel/v1.0/data/Volume
	Discount/Volume
	Discount/{id}/Volume
	Discount Item",
	"source_request_type":
	"GET",
	"query_parameter":
	"@pathParam(id)",
	"query_value_json_path":
	"$.Id",
	...
	}</pre>
		Used in query map to pass value for a path parameter key.

Supported Templates

The API configuration allows you to define template files for the source API calls. They are configured as part of the mapping file definition.

Templates used by Siebel CRM

Table 6-3 lists template files that are referenced by the source API configuration during the migration event.

Table 6-3 Templates used by Siebel CRM

Template	Description
catalogPOSTTemplate.json	Used to retrieve catalog from Siebel.
attributePOSTTemplate.json	Used to retrieve attributes from Siebel.
classPOSTTemplate.json	Used to retrieve product class from Siebel.
plPOSTTemplate.json	Used to retrieve product lines from Siebel.
productPOSTTemplate.json	Used to retrieve product from Siebel.
promotionPOSTTemplate.json	Used to retrieve promotion from Siebel.
skuPOSTTemplate.json	Used to retrieve Smart Part Number from Siebel.
adjustmentGroupPOSTTemplate.json	Used to get Product-Based Adjustment for discount matrices from Siebel.
bulkPromotionPostTemplate.json	Used to get bulk of promotions from Siebel.
productComponentsPOSTTemplate.json	Used to get the CP full structure from Siebel.
recommendationRulePOSTTemplate.json	Used to get recommendation rules from Siebel.
promotionPostIdsTemplate.json	Used during nested job processing for Siebel promotion.

Table 6-3 (Cont.) Templates used by Siebel CRM

Template	Description
ProductPostIdsTemplate.json	Used during nested job processing for Siebel products.
catalogIDTemplate.json	Used to retrieve Catalog ID from Siebel.
productClassIDTemplate.json	Used to retrieve Product Class ID from Siebel.
productLineIDTemplate.json	Used to retrieve Product Line ID from Siebel.
productIDTemplate.json	Used to retrieve Product ID and Promotion ID from Siebel.

Table 6-4 Templates used by Siebel Migration

Template	Description
childProductCallsPOSTTemplate.json	Used to retrieve child products of a bundle offer from Siebel.

Table 6-5 Templates used by Siebel Publish

Template	Description
productPOSTTemplate.json	Used to retrieve child products of a bundle offer from Siebel.
promotionPOSTTemplate.json	Used to retrieve promotion from Siebel.
productComponentsPOSTTemplate.json	Used to get the CP full structure from Siebel.
skuPOSTTemplate.json	Used to retrieve Smart Part Number from Siebel.

Table 6-6 Supported Application Constants

Name	Sample Payload	Description

@UUID	<pre>"someField": {
	"type": "string",
	"x-oracle-map-data": {
	"json_path": "$.dummyField",
	"mapType": "TO_FIELD",
	"default": "@UUID"
	}
	}</pre>
		Returns an universally unique identifier.

Table 6-6 (Cont.) Supported Application Constants

Name	Sample Payload	Description
@datetime	"Current DateTime": { "type": "string", "x-oracle-map-data": { "json_path": "$.dummyField", "mapType": "TO_FIELD", "default": "@datetime" } }	Returns the current DateTime in GMT.

Managing Mapping File Versions

Oracle ships the three seeded grammar files:

- One-time migration: The **siebel_launch.json** file that contains the mapping with source as Siebel and target as Launch.
- Publish to Siebel: The **launch_siebel.json** file that contains the mapping with source as Launch and target as Siebel.
- Publish to PDC: The **launch_pdc.json** file that contains the mapping with source as Launch and target as PDC.

You can retrieve the seeded mapping file using the following command:

```
GET - {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
```

> **Note**
>
> FA_APIGW in the above endpoint is FA (Fusion Application) API Gateway.

Headers to be passed:

- X-Source-System : <<SourceName>>
- X-Destination-System : <<DestinationName>>
- X-Seeded-Version : true

Any additional extensions that you need to bring into the mapping is done using the versioning of the mapping file using the seeded grammar file as the starting point.

> **Note**
>
> The seeded files should never be modified as all release updates make it into these files.

(Optional)To version the grammar files, you can pass a unique version as part of Mapping APIs using the X-Version header. If the header is not passed, the default version of the mapping is used.

After creating the grammar files using different versions, pass the desired version in the header X-Version to run the migration job. The migration and publish job will identify the grammar file by the version and will use it to run the job. For more information on how to extend using the seeded grammar files, see Siebel Extensions Handling.

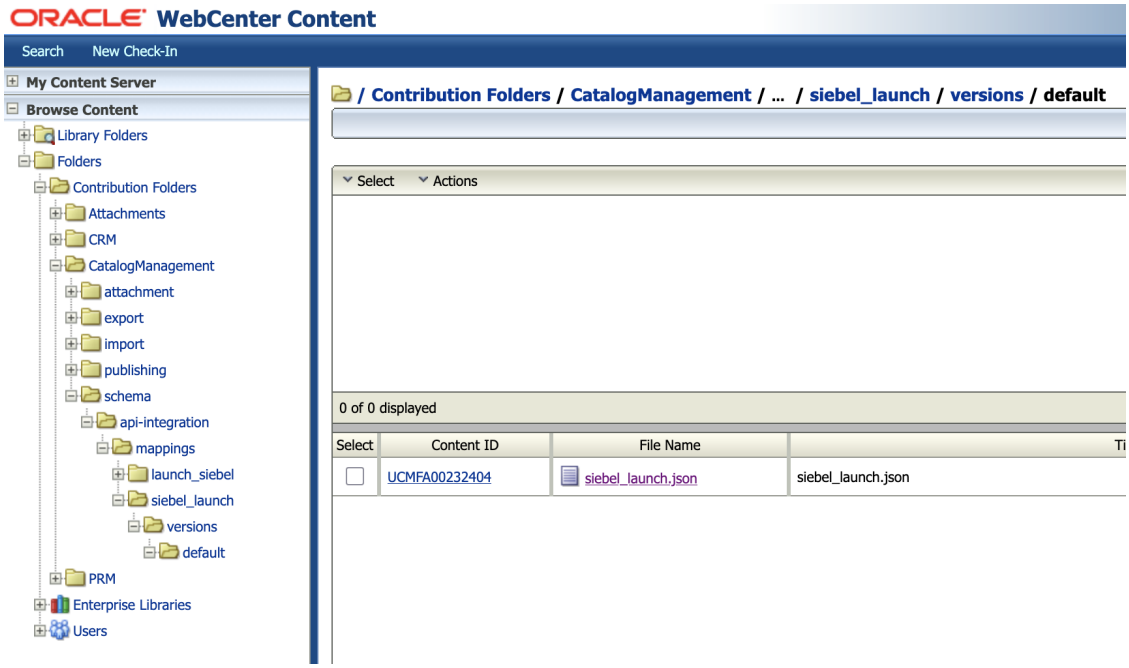
Mapping File Upgrade with Customer Customization

Oracle ships the seeded mapping file in each release on top of which a customer can add their extensions and update it in Fusion "Content Store" for use. For more information, see Handling Extensions.

> **Note**
>
> Any mapping extension done by you in the current release needs to be redone again on top of the seeded file shipped by Oracle for (current + 1) release once the environment upgrade is complete. This is a manual task that needs to be done carefully following the procedure mentioned below.

Assume that customer added few extensions to the mapping file in the current release and updated it in Fusion Content Store. Their changes would be part of the default version folder in Fusion "Content Store". See Figure 6-1 for the "Content Store" folder structure.

Figure 6-1 Default Version Folder in Fusion Content Store



Let's say environment is upgraded to next available release (current + 1) with LaunchX services. In this case, Oracle would again ship the seeded mapping file corresponding to (current + 1) release and any mapping extension done in the previous release needs to be redone again on the top of the seeded file shipped by Oracle for (current + 1) release. Follow the below steps to make the changes seamlessly.

- Get the mapping file with extension changes done in current release from Fusion "Content Store" default version folder using below endpoint.
  `GET - {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema`
  - Headers to be passed:
    - X-Source-System : <<sourceName>>
    - X-Target-System : <<targetName>>

> **Note**
>
> FA_APIGW in the endpoint is Fusion Application API Gateway.

- Verify the response of the above GET endpoint. It should have all the mapping extension done in current release. Save the response of the above GET endpoint to a file in your local machine.
- Create a folder with another version say v1.0 in Fusion "Content Store" having backup of the above saved mapping file with added extensions using below endpoint:
  `PUT - {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema`
  - Headers to be passed:
    - X-Source-System : <<sourceName>>
    - X-Target-System : <<targetName>>

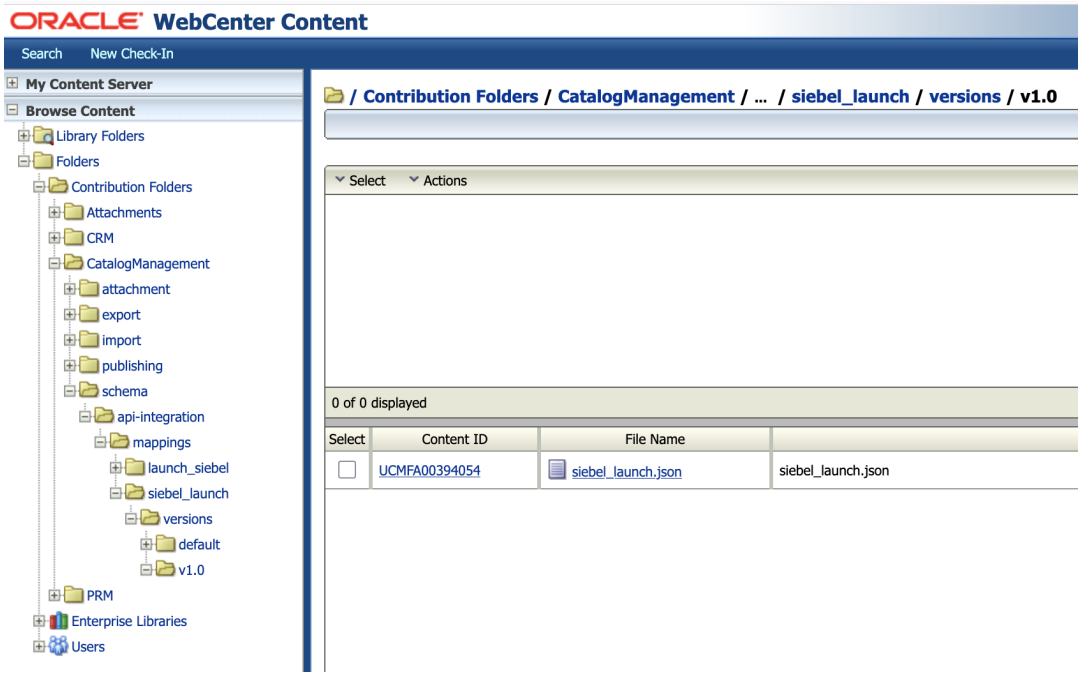
* X-Version : v1.0

– Body to be passed:

* File : <<file saved in earlier step to be uploaded here>>

> **Note**
>
> FA_APIGW in the endpoint is Fusion Application API Gateway.

Figure 6-2 Version 1.0 Folder in Fusion Content Store



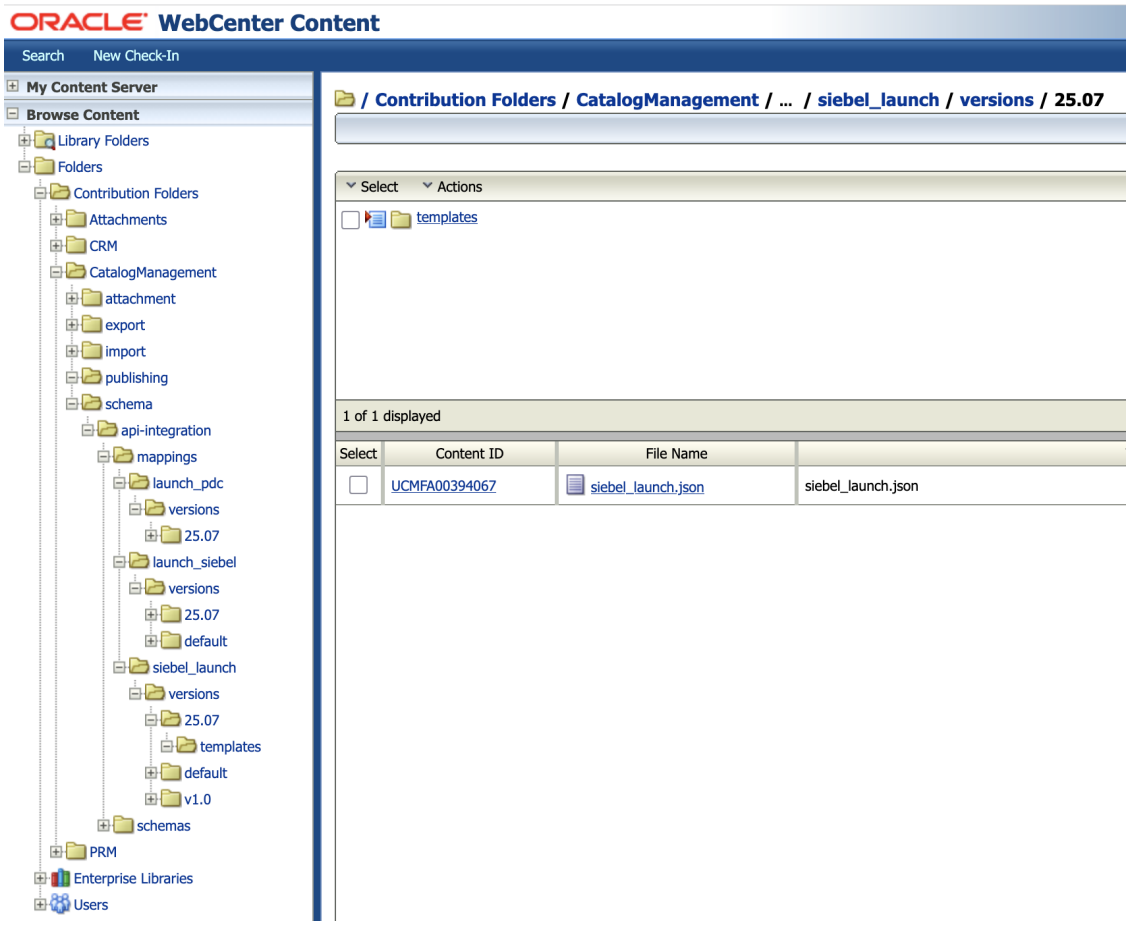
- Invoke the below endpoint and create mapping files and templates shipped by Oracle for next release (current + 1) in Fusion "Content Store". The response status code for endpoint should be 200 with no response body. See Figure 6-3 for Content Store structure for changes done after the endpoint is run successfully. This endpoint creates a new folder under versions with name taken from VERSION file shipped as part of mapping service. The release version is stored in the VERSION file. For example 25.01, 25.04, 25.07, 25.10, and so on.

  `Endpoint : {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/loadSeedFiles`

  Method Type: GET

  > **Note**
  >
  > FA_APIGW in the endpoint is Fusion Application API Gateway.

Figure 6-3 Version 25.07 Folder in Fusion Content Store



Take the Oracle shipped mapping file as part of (current + 1) release present in its corresponding version folder in the Content Store created in the above step to your local machine. Update it manually with the extension changes done in current release present in the mapping file part of default version folder. Save the file in your local machine once all the changes are completed. Note that final file content would be content of (current + 1) release mapping file shipped by Oracle including the mapping extension attempted in current release (added manually).

Update the saved file in the above step to Content Store default folder with the below endpoint.

```
PUT - {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
```

- Headers to be passed:
  - X-Source-System : <<sourceName>>
  - X-Target-System : <<targetName>>
- Body to be passed:
  - File : <<file saved in the earlier step to be uploaded here>>

> **Note**
>
> FA_APIGW in the endpoint is Fusion Application API Gateway.

Once the file is updated in Fusion Content Store, do a get call and verify the file content to ensure it has been updated correctly in Content Store.

`GET - {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema`Headers to be passed:

- X-Source-System : <<sourceName>>
- X-Target-System : <<targetName>>

> **Note**
>
> FA_APIGW in the endpoint is Fusion Application API Gateway.

Another way to verify if the file is updated in Content Store is by logging in to Fusion "Content Store" in a web browser. Download the file from Content Store and verify the file content.

URI for accessing Fusion Content Store:

{{FA_HostName}}/cs

> **Note**
>
> FA_HostName in the above endpoint is Fusion Application Host Name.

Once the mapping file content is verified, start migrating/publishing the data to/from Launch.

Siebel Extensions Handling

It is quite common for Siebel customers to have extensions to Product definitions to meet their business needs.

Mapping definitions are preconfigured for both publish and migration jobs with the required template files for the out-of-the-box Siebel fields. To accommodate extensions to entities, the seeded mapping must be updated, or new mapping configurations can be created using the REST APIs.

All extensions to Launch for the identified Siebel extensions to product definitions must be done using Launch Extensibility. See *Launch Cloud Service Implementation Guide* for more details. For example, the extensions are made to productOfferingOracle.yml file to create productOfferingCustomerName.yml file. Once the schema between the two applications are in place, the next step is to incorporate these extensions into the Migration and Publish grammar files.

The steps to add extension in grammar files:

Migration Mapping File

1. Get the seeded mapping file for migration to Launch. The seeded mapping file can be retrieved by using GET grammar call with details mentioned below. For Siebel migration to Launch, the X-Source-System header should be “siebel” and X-Destination-System should be “launch”.

   Endpoint : {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema

   Method Type: GET

Headers to be passed:

- X-Source-System : siebel
- X-Destination-System : launch
- X-Seeded-Version : true

2. Once the grammar file is retrieved, the next step is to make the mapping edits to extend the base schema. Open the mapping file in any text editor of your choice.

   Let's say we extend the Launch ProductOfferingOracle resource for a simple offer and assume name of the extended resource is ProductOfferingCustomer with extended fields as rating, a free-form text and priceType being a bounded picklist.

   Below steps to be formed in mapping file:

   a. Get the name of base schema in the mapping file responsible for preparing transformed payload for simple offer. The base schema generally has a relevant name in mapping file schemas section.

   b. Define a new schema in the mapping file extending the base schema identified in step 1.

   > **Note**
   >
   > Provide a unique name to the new schema in the mapping file. Else, the following error is displayed while updating the file in UCM (Content Store): "The File Content is not adhering valid OpenAPI standard".

   c. Use the below sample snippet to define a new schema extending the OOB schema with the additional extended properties. We use Swagger OpenAPI allOf construct to extend existing schema.

```
"ProductOfferingCustomerName": {
    "title": " ProductOfferingCustomerName",
    "type": "object",
    "allOf": [
        {
            "$ref": "#/components/schemas/<base-schema-name-in-mapping-
file-to-be-extended>"
        },
        {
            "properties": {
                "@type": {
                    "type": "string",
                    "x-oracle-map-data": {
                        "json_path": "$.dummyField",
                        "mapType": "TO_FIELD",
                        "default": "<@type-value-of-new-extended-
resource-in-launch>"
                    }
                },
                "resourceExtension": {
                    "type": "object",
                    "title": "resourceExtension",
                    "x-oracle-map-data": {
                        "json_path": "$",
                        "mapType": "TO_OBJECT"
```

```
                        },
                        "properties": {
                            "rating": {
                                "type": "integer",
                                "x-oracle-map-data": {
                                    "mapType": "TO_FIELD",
                                    "json_path": "<json-path-to-map-to-
siebel-system-field>"
                                }
                            },
                            "Price Type": {
                                "type": "string",
                                "x-oracle-map-data": {
                                    "json_path": "<json-path-to-map-to-
siebel-system-field>",
                                    "mapType": "TO_FIELD",
                                    "value_map": {
                                        "pre_transform": false,
                                        "key_value_set": {
                                            "One-Time":
"ONE_TIME_PRICE_PLAN",
                                            "Recurring":
"RECURRING_PRICE_PLAN",
                                            "Usage": "USAGE_PRICE_PLAN"
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        ]
}
```

Make the following changes to the above sample snippet:

- $ref in allOf needs to point to the extended base schema.
- Override base schema @type field to the newly extended schema @type value.
- Write the JsonPath for extended fields in the new schema to map them to their corresponding fields in Siebel stitched data.

3. Identify all the target_request_spec in the mapping file responsible for doing transformation for identified base schema and change its component field to point to new schema. Note that target_request_spec component field controls the selection of schema to be picked for transformation.
4. Update the modified mapping file in UCM (Content Store) using the following endpoint:

   Endpoint : {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema

   Method Type: PUT

   Headers to be passed:

   - X-Source-System : siebel
   - X-Destination-System : launch

Body to be passed:

File <File-to-be-uploaded>

5. Once the file is updated in UCM (Content Store), do a get call to verify if the file is updated correctly in UCM.

   Endpoint : {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema

   Method Type: GET

   Headers to be passed:

   - X-Source-System : siebel
   - X-Destination-System : launch

   Another way to verify if the file is updated in UCM is by logging in to UCM (Content Store) in web browser. Download the file from UCM and verify the file content.

6. Once the mapping file is updated and content is verified, start migrating the data from Siebel to Launch.

Publish Mapping File

1. Get the seeded mapping file for publishing to Siebel. The seeded mapping file can be retrieved by using GET grammar call with details mentioned below. For publishing to Siebel, the X-Source-System header should be “launch” and X-Destination-System should be “siebel”.

   Endpoint:

   ```
   {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
   ```

   Method Type: GET

   Headers to be passed:

   - X-Source-System : launch
   - X-Destination-System : siebel
   - X-Seeded-Version : true

   Curl command:

   ```
   curl --location 'https://gxl4jyzzuv5zu2nhmsb47apvfy.apigateway.us-
   ashburn-1.oci.customer-oci.com/mappingSchema/cx/industry/apiIntegration/v1/
   mappingSchema' \
   --header 'X-Source-System: launch'\
   --header 'X-Destination-System: siebel'\
   --header 'X-Seeded-Version: true'\
   --header 'Authorization: ••••••'\
   --data ''
   ```

2. Once the grammar file is retrieved, make the mapping edits to extend the base schema. Open the mapping file in any text editor of your choice.

   Below steps to be formed in mapping file:

   a. Identify the schema to be extended from mapping file. Generally, the schema will be having a relevant name and it can be found in the schemas section of mapping file.

   b. Define a new schema in the mapping file to extend the schema identified in step a.

> **Note**
>
> Provide a unique name to the new schema in the mapping file. Else, the following error is displayed while updating the file in UCM (Content Store): “The File Content is not adhering valid OpenAPI standard”.

c. Identify the position of the fields/references to be extended. This should be done after reviewing Siebel API contract. The respective Siebel API contract can be retrieved using describe API endpoint from Siebel. Maintain the same structure while extending the fields in the new schema.

d. Add the fields to be extended in the newly created schema.

> **Note**
>
> In the extended schema, add only the fields/references which are affected in the extension. The rest of the fields should be inherited from the base schema using Swagger OpenAPI ‘allOf’ construct.

Following are some examples for extending schema.

Example 1: Adding LOV to an existing field

In the following base schema ‘Product’, for the field paymentType, only 2 values are supported – ‘Prepaid’ and ‘Postpaid’. Now the customer wants to extend the schema so that this field will support an additional payment type value which is ‘Hybrid’.

Base schema:

```
{
  "Product": {
    "title": "Product",
    "type": "object",
    "properties": {
      "Payment Type": {
        "key_value_set": {
          "PREPAID": "Prepaid",
          "POSTPAID": "Postpaid"
        }
      },
      "Card Type": {
        "key_value_set": {
          "DEBIT": "Debit",
          "CREDIT": "Credit"
        }
      }
    }
  }
}
```

Extended schema:

```
    "ProductExtended": {
       "title": "ProductExtended",
       "type": "object”,
```

```
        "allOf": [
          {
            "$ref": "#/components/schemas/Product"
          },
          {
          "properties": {
                 "Payment Type": {
                        "key_value_set": {
                                                "PREPAID": "Prepaid",
                                                "POSTPAID": "Postpaid"
                                                "HYBRID": "Hybrid"
                                             }
                                          }
                                   }]

                        }
```

Example 2: Adding new field

Below is one sample snippet extending the base schema and adding the extended field under [SWI Product Definition VBC]. We use Swagger OpenAPI allOf construct to extend existing schema.

```
     "ProductOfferingCustomer": {
        "title": " ProductOfferingCustomer",
        "type": "object",
        "allOf": [
          {
            "$ref": "#/components/schemas/<base-schema-name-in-mapping-
file-to-be-extended>"
          },
          {
            "properties": {
              "SiebelMessage": {
                "type": "object",
                "title": "SiebelMessage",
                "x-oracle-map-data": {
                  "json_path": "$",
                  "mapType": "TO_OBJECT"
                },
                "properties": {
                  "ListOfSWIProductIntegrationIO": {
                    "type": "object",
                    "title": "ListOfSWIProductIntegrationIO",
                    "x-oracle-map-data": {
                      "json_path": "$",
                      "mapType": "TO_OBJECT"
                    },
                    "properties": {
                      "SWI Product Integration VBC": {
                        "type": "array",
                        "x-oracle-map-data": {
                          "json_path": "$",
                          "mapType": "TO_ARRAY"
                        },
                        "items": {
```

```
                                "type": "object",
                                "properties": {
                                  "ListOfSWI Product Definition VBC": {
                                    "type": "object",
                                    "title": "ListOfSWI Product Definition
VBC",
                                    "x-oracle-map-data": {
                                      "json_path": "$",
                                      "mapType": "TO_OBJECT"
                                    },
                                    "properties": {
                                      "SWI Product Definition VBC": {
                                        "type": "array",
                                        "x-oracle-map-data": {
                                          "json_path": "$",
                                          "mapType": "TO_ARRAY"
                                        },
                                        "items": {
                                          "type": "object",
                                          "properties": {
                                            "SiebelExtendedField": {
                                              "type": "string",
                                              "x-oracle-map-data": {
                                                "json_path": "<json-path-to-
map-to-launch-system-field>",
                                                "mapType": "TO_FIELD"
                                              }
                                            }
                                          }
                                        }
                                      }
                                    }
                                  }
                                }
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
              ]
            }
```

Make the following changes to the above sample snippet:

- $ref of allOf swagger construct needs to point to the extended base schema.
- Write the JsonPath for newly extended fields in siebel to map them to their corresponding fields in Launch stitched data.

> **Note**
>
> Note In the above example, a new field SiebelExtendedField is added inside SWI Product Definition VBC. Existing fields will be inherited from base schema, only the tree hierarchy needs to be maintained.

3. Identify the target_request_spec's from the mapping file which are impacted by the schemas, which are extended, and follow the steps given below based on the requirement.

   a. Change the eligibility_check_json_path and source_data_path fields of identified target_request_spec to include the new extended resource.
   A sample json path is given below:

   JsonPath before Extension: "`$..projectItems[?(@.isBundle == 'false' && @.['@referredType'] IN [‘<@referredType of schema to be extended>’])]`"

   JsonPath after Extension: "`$..projectItems[?(@.isBundle == 'false' && @.['@referredType'] IN [‘<@referredType of the extended schema>’])]`".

   > **Note**
   >
   > If we want both the schemas, the original schema as well the extended, we should keep both in the jsonPath as given below:
   > If we want both the schemas, the original schema as well the extended, we should keep both in the jsonPath as given below:
   >
   > JsonPath after Extension: "`$..projectItems[?(@.isBundle == 'false' && @.['@referredType'] IN [[‘<@referredType-of-schema-to-be-extended>’ ,‘<@referredType-of-the-extended-schema>’])]`".
   >
   > Most of the cases, we keep both to ensure backward compatibility.

   b. Additionally, change the component field of target_request_spec to point to the newly extended schema in mapping file. Note that target_request_spec ‘componentName’ field controls the selection of schema to be picked for transformation.

4. Update the modified mapping file in UCM using the following endpoint:
   Endpoint:

   ```
   {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
   ```

   Method Type: PUT

   Headers to be passed:

   - X-Source-System : launch
   - X-Destination-System : siebel

   Body to be passed:

   File <File-to-be-uploaded>

5. Once the file is updated in UCM (Content Store), do a get call to verify if the file is updated correctly in UCM.
   Endpoint:

   ```
   {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
   ```

Method Type: GET

Headers to be passed:

- X-Source-System : launch
- X-Destination-System : Siebel

6. Once the mapping file is updated and verified, start publishing the data from Launch to Siebel.

Supported Universal Content Management (UCM) Calls

UCM is the content store where the grammar files are stored in folders with versioning of the changes. You can create a new mapping file, retrieve the latest to make updates and upload the changes made..

Create Mapping OAS File

To create a mapper file, use the below request: similarly for others:

```
POST - {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
  • Headers to be passed:
     o X-Source-System <<sourceName>>
     o X-Target-System <<targetName>>
  • Body to be passed
    o File. <<file to be uploaded>>
```

> **Note**
>
> FA_APIGW in the above endpoint is FA (Fusion Application) API Gateway.

For more details on header key usage, see Supported Functions in this guide.

Update Mapping OAS File

To update a mapper file, use the below request:

```
PUT - {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
• Headers to be passed:
   o X-Source-System <<sourceName>>
   o X-Target-System <<targetName>>
• Body to be passed
   o File. <<file to be uploaded>>
```

> **Note**
>
> FA_APIGW in the above endpoint is FA (Fusion Application) API Gateway.

Create Template File

```
POST – {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/template
• Headers to be passed
     o X-Source-System- <<SourceName>>
     o X-Target-System- <<targetName>>
• Body to be passed
     o File- <<file to be uploaded>>
```

Update Template File

```
PUT – {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/template
• Headers to be passed
     o X- Source-System - <<SourceName>>
     o X-Target-System - <<targetName>>
• Body to be passed
     o File - <<file to be uploaded>>
```

Troubleshooting Integration Errors

This section describes the common issues that could occur either during catalog sync or during publish or migration.

Table 6-7 Troubleshoot Integration Errors

Error Scenario	Error Description	Troubleshooting Tips
Publish is successful, however Product is not created in Siebel CRM.	No error in the UI will be shown.	Offers with at least one offer only will be published to Siebel CRM. Ensure that the offer has pricing defined.
Publish does not get initiated	HTTP 404 error in browser console	Ensure that Destination is configured correctly.
Publish fails with error	Processing Failure	Use REST APIs in test mode to analyze further.
Publish status of the initiative does not change even after running for a long time.	No visible error reported.	PATCH the publish job as FAILED using REST API and then republish the initiative.
Gateway timeout in test mode.	Launch REST APIs returns error when run in test mode.	Publish - filter the results by adding the header Component-Name Migration – Run the migration job in Async- Mode.
Migration fails with error "A bundle product offer must be provided when the isBundle is true"	Not Applicable	Siebel CRM supports Customizable Product without having any component products. But Launch does not allow to create a Bundle Offer without adding any component products to it.

Table 6-7 (Cont.) Troubleshoot Integration Errors

Error Scenario	Error Description	Troubleshooting Tips
Migration fails with error “The subresource characteristic entity with the (xyz) value combination already exists. Enter unique values.”	This issue comes up when an attribute is used more than once in the product class. Here xyz in error msg is the attribute name being used more than once in a class.	In Siebel, attribute is a top-level entity and a class can have same attribute definition more than once with different name. But Launch does not allow to use same attribute definition more than once. Hence migration fails.

7

Extending the Mapping with Custom Services

Launch has productized publishing capabilities with Siebel, BRM - wherein we follow a low-code approach to publish into runtime applications. This has been made possible using the OAS mapper capability (using JSON path function to query for an element within JSON data). This allows you to map entities between source (Launch) and target (runtime systems) using the provided set of java classes (standard functions) for transformation such as valueMap, defaultValue, splitString, conditionalValueMap, and so on. However, there might arise a need to add your own mapping functions to support any custom transformation logic that might be required which is not available in the product. This improves the business agility to distribute catalog definitions by adding your business specific mapping functions.

Integration of Launch, Siebel, and BRM requires creation and upload of mapper file that will point to your custom mapping. A default mapper file is provided by Launch. You can update the same file with invocation calls to the custom mapping function. For more detail, see Integration with Siebel (CRM) and Integration with PDC (BRM) . This integration of external mapping function supported in the mapping file involves a series of steps. These include declaring the function signature within the mapping file, setting up the wiring, and developing the third-party function service, among other components.

Overview

This chapter outlines the Integration of Third-Party Mapping Function service to support extension of current productized mapping capabilities supported by mapper file. This chapter also provides detailed step-by-step instructions for integrating a third-party mapping function service.

Assumptions: The external function service developed by Customers, ensures high availability and low latency. The function service implementation should expose APIs as described in this document.

> **Note**
>
> Raise a Service Request with Oracle Support before you start adding external mapping services for Integration to Siebel CRM and BRM.

Purpose

The purpose of this integration is to enable Launch to interact with customer developed mapping service having preTransform and transform API based on swagger definition. This integration provides enhanced functionality by allowing the Launch Integration microservices to extend the mapping capabilities currently offered by mapper file.

Scope

This chapter covers the entire process from initial configuration in Fabric to final testing in Launch, including API setup, authentication configuration, and Launch-specific settings.

Prerequisites

Before beginning the implementation, ensure you have the following:

- Access to the CXIF Host environment (e.g., https://your-cxif-host-example.com).
- Customers Admin privileges for running/admin APIs.
- Authentication credentials for the third-party Function Service (OAuth2, Basic Auth, or OCIHttpSignature).
- Familiarity with RESTful APIs and JSON.
- Downloading Third Party Function Service Swagger. See Appendix for more details.
- Access to curl or Postman for API testing.
- Knowledge of the specific service being integrated.
- Knowledge of mapper file, its constructs, and its usages.

Related Guides

Table 7-1 contains information about other useful sources of information for the integration process.

Table 7-1 Related Guides for External Mapping Services Integration

Reference	Description
Implement CX Industries Framework	Describes the setup and implementation of the CX Industries Framework required for Launch integrations.

System Architecture Overview

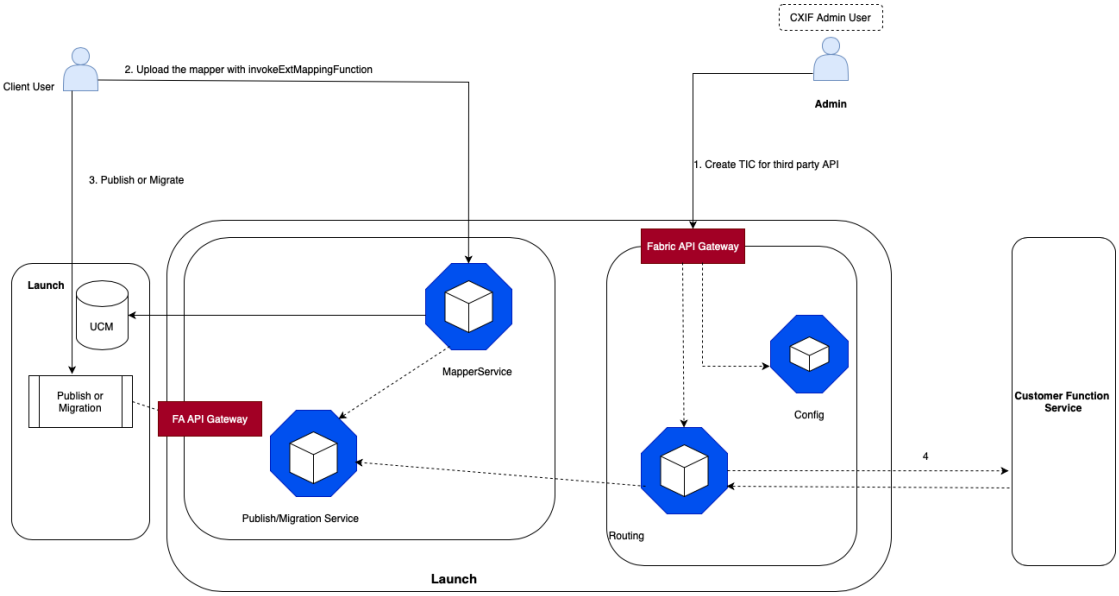
The integration involves three main components:

1. **Fabric**: Acts as the API management and routing layer. Fabric also have Integration microservices for uploading the mapper file and perform publish or migration.
2. **Client's Third-party Function Service**: The third-party mapping function service provided and hosted by client.
3. **Launch**: The UI for Publish and Migration.

The flow of data is as follows:

Launch > Third-party Mapping Function Service

Figure 7-1 Integrating Launch, Fabric, and the Third Party Function Service



1. The diagram illustrates the architecture for integrating Launch, Fabric, and the third party function service. Assuming the Third party Function Service is ready, the process begins with creating a TIC or Connection Descriptor in the Config microservice. This TIC specifies authentication method, credentials, and host information of the Third Party Function Service.
2. The mapper file with the necessary mapping extension changes is uploaded. The file is stored in UCM.
3. The user publishes or migrates the project.
4. The Customer Function Service is called and returns the appropriate response.

Detailed Implementation Steps

Configuring Fabric

For Launch to communicate with RMS, and for RMS to make outbound calls to the “Third Party Function Service,” certain configurations must be made in the CXIF fabric cluster to mediate the flow. This configuration should be completed using the CXIF Admin APIs. More details about each individual service can be found in the Related Guides section.

Create a New Connection Descriptor (TIC)

Connection Descriptor or TIC is where one would mention the host of the third-party function service deployed. Also, the authentication mode and the authentication secrets are to be passed in the request while creating TIC.

Supported authentication modes and sample requests for the same are provided below for reference.

> **Note**
>
> Do not modify the fields **endpoint-name**, **type**, and **system-descriptor**. You can modify other fields as per your implementation

Sample Request:

```
POST https://{CXIFHost}/admin/connectionDescriptors
{
  "endpoint-name": "ThirdPartyMappingFunction",
  "endpoint-url": "https://your-host-example.com/",
  "fabric-facing-auth": {
    "oidc-client-credentials": {
      "client-id": "your-client-id",
      "client-secret": "your-client-secret",
      "identity-uri": "https://your-identity-provider.com/oauth2/v1/token",
      "scope": "your-scope"
    }
  },
  "type": "external",
  "system-descriptor": "thirdpartymappingfunction-ttd"
}
```

endpoint-url field should be the host of the service which interacts with the underlying third party function service.

fabric-facing-auth field should be where we decide the mode of authentication and the credentials for authenticating.

Replace placeholders with your actual third party mapping function service endpoint and OAuth2 credentials. Currently fabric supports two types of authentications. BasicAuth and OAuth2 (only client credentials is supported).

The sample payload for each type is as mentioned below.

Table 7-2 Sample Payload

Authentication Type	Sample Payload

Basic Auth	{
	"system-descriptor": "thirdpartymappingfunction-ttd ",
	"endpoint-name": "ThirdPartyMappingFunction",
	"endpoint-url": "https://
	thirdpartyfunctionservice.dev.com/",
	"fabric-facing-auth":{
	"basic":{
	"username": "admin",
	"password": "*password*"
	}
	},
	"type": "external"
	}

Table 7-2 (Cont.) Sample Payload

Authentication Type	Sample Payload

OAuth2	{
	"endpoint-name": "ThirdPartyMappingFunction",
	"endpoint-url": "https://
	thirdpartyfunctionservice.dev.com/",
	"fabric-facing-auth": {
	"oidc-client-credentials": {
	"client-id":"48eb39a9a7cb4bc0b7761ebb8d3ada97",
	"client-secret":"adt2cdde-6c94-4be2-b525-
	fff575a9c3fc",
	"identity-uri": "https://
	idcs-322c58839e042ad2.identity.oraclecloud.com/oauth2/v1/
	token",
	"scope": "https://n6jfpge6uqfrum.apigateway.us-
	ashburn-1.oci.customer-oci.comurn:opc:resource:consumer::"
	}
	},
	"type": "external",
	"system-descriptor": "thirdpartymappingfunction-ttd "
	}]

Creating GKR (Gate Keeping Rule)

Once TIC is created, a default GKR will get created with endpoint-name **ThirdPartyMappingFunction** after 10 seconds.

Use the below get call to get the id of created GKR.

```
GET {{Fabric_APIGW}}/admin/gatekeepingRules
```

Response of this get call will be the list of GKR.

Sample Response

```
[
    {
        "endpoint-name": "ThirdPartyMappingFunction",
        "rule-name": "Generated gatekeeping rule for endpoint tmf632",
        "destination-selection": [
            {
                "api-id": "orclfunc-100",
                "api-version": "v1",
                "criteria": [
                    {
                        "rank": 1,
                        "resource-ids": [
                            "transform",
                            "preTransform"
                        ]
                    }
                ]
            }
        ],
        "id": "gkr-internal-rest-1234 "
    },
    {
```

```
        "endpoint-name": "pdc-test2",
        "rule-name": "Generated gatekeeping rule for endpoint pdc-test2",
        "id": "gkr-pdc-test2zgj5k"
    }
]
```

From the list of responses, the default GateKeepingRule for **ThirdPartyMappingFunction** can be considered. Use the id from the default GateKeepingRule of **ThirdPartyMappingFunction** to make the PUT call as shown below:

```
{{Fabric_APIGW}}/admin/gatekeepingRules/gkr-func7n7fw
```

Sample Payload

```
{
    "endpoint-name": "ThirdPartyMappingFunction",
    "rule-name": "Generated gatekeeping rule for endpoint func",
    "destination-selection": [
        {
            "api-id": "orclfunc-100",
            "api-version": "v1",
            "criteria": [
                {
                    "rank": 1,
                    "resource-ids": [
                        "transform",
                        "preTransform"
                    ]
                }
            ]
        }
    ],
    "id": "gkr-func7n7fw"
}
```

The POST call above will create the GKR.

> **Note**
>
> Only the id field of the above payload needs to be changed and copied from GET call output. Other fields will remain unchanged.

Validating the Connection and Testing the API

After configuring Fabric, it's crucial to test the connection before proceeding to Launch configuration.

PreTransform API Test - Refer Appendix : Downloading swagger file.

Prepare a POST request:

```
https://<fabricHost>/api/01/apiIntegration/v1/preTransform with payload.
```

Sample Payload:

```
{
  "functionName": "customExternalMappingFunction",  // This function should be
implemented by third party mapping function service.
  "inputJson": "{ “name” : “IPhone”, “id” : “2222”}” // This input is received from
making source API calls and will be input for customExternalMappingFunction.
```

```
  "paramList": [red],
  "contextParameters": [
    {
      "Name": "JobId",
      "Value": "1234"
    }
  ]
}
```

Send the request and analyze the response. You should receive result in the format:

```
{
"result": {
"valueType": "object",
"value": "{}”
}
}
```

For Pretransform, valueType can be either an object or an array, and value should be of same data type as mentioned in valueType.

Transform API Test - Refer Appendix : Downloading swagger file

Prepare a POST request:

```
https://<fabricHost>/api/01/apiIntegration/v1/transform with payload
```

Sample Payload:

```
{
  "functionName": " customExternalMappingFunction ",  // This function should be
implemented by third party mapping function service.
  "inputJson": "{ “name” : “IPhone”, “id” : “2222”}” // The input is received from
making source API calls and will be input for customExternalMappingFunction.
  "paramList": [],
  "contextParameters": [
    {
      "Name": "JobId",
      "Value": "1234"
    }
  ]
}
```

Send the request and analyze the response. You should receive result in the format:

```
{
"result": {
"valueType": "object",
"value": "{}”
}
}
```

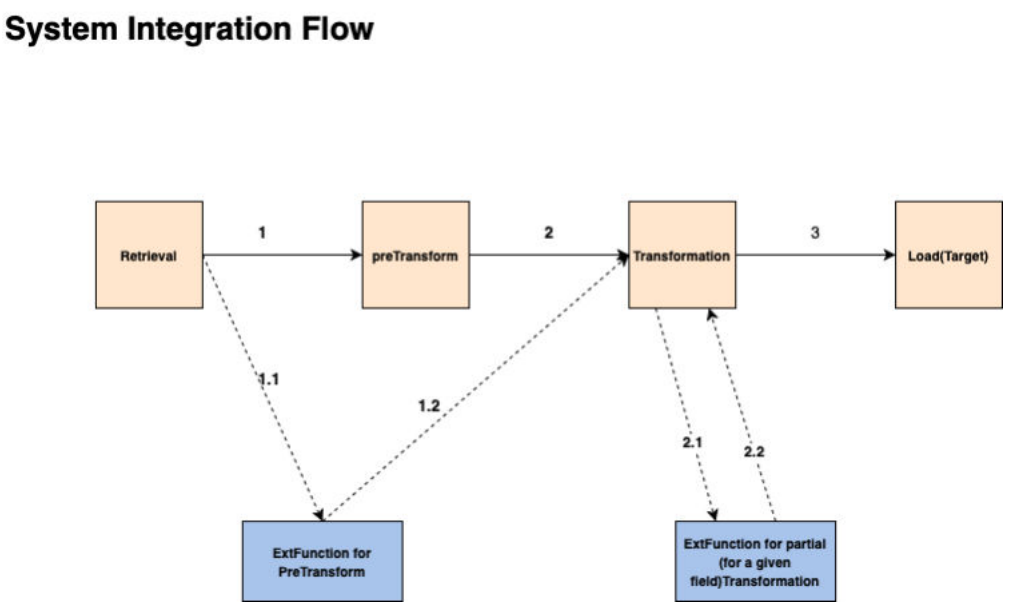
For transform, valueType can be either an object, an array, integer, string, or number and value should be of same data type as mentioned in valueType.

Changes in Mapper File

Maintaining versions and uploading Mapping File procedure can be referred from Integrate Launch with Siebel CRM and Pricing Design Center (BRM) guide. This section only describes the changes to be done in Mapping File.

Mapping File Changes for Transform and PreTransform

Figure 7-2 System Integration Flow



In the mapper file, source_request_spec defines the procedure of retrieving source data. After the retrieval of source data, pre transformation happens (see 1 in Figure 7-2). There is a field in target_request_spec called select_json_path, whose value defines the pre transformation logic. The value of select_json_path can be jaywayjson path or function or internal function.

For example, if "select_json_path": "$", this means that the entire payload, starting from the root, will be selected for transformations.

To further enhance the capability of select_json_path, a user can use his own defined functions (third-party functions) as a value of select_json_path (see 1.1 in Figure 7-2), the below signature can be used. Once this signature is processed in runtime, the preTransform API is invoked with appropriate parameters. Refer PreTransform Test Sample Payload.

" select_json_path ": "@invokeExtFunction(customExternalMappingFunction, PRETRANSFORM, arg1, arg2…)"

The signature is explained below:

@**invokeExtFunction** is the signature to be used for calling third-party function service REST API.

customExternalMappingFunction is the function name that needs to be invoked by third party function service.

PRETRANSFORM is a keyword that indicates that function is called on the context of "pre-transformation" and respective API will get called.

arg1 and **arg2** are optional parameters. There can be infinite optional parameters.

Once pre transformation is completed, transformation begins (see 2 in Figure 7-2). Component schema is the structure of payload that should be sent to target as payload to respective exposed API. Component schema contains different fields. Mostly fields are evaluated using

jayway json path, but function and oracle internal functions are also supported. The third-party function service can be used with component fields as well (see 2.1 and 2.2 in Figure 7-2).

The following sample demonstrates how to specify function signature to invoke third party function service for transformation. Once this signature is processed in runtime, the transform API is invoked with appropriate parameters. Refer Transform Test Sample Payload.

```
"productName": {
  "type": "string",
  "description": "Description of this Employee",
  "x-oracle-map-data": {
    "json_path": "@invokeExtFunction(customExternalMappingFunction, TRANSFORM,
params...)"
  }
}
```

@invokeExtFunction is the signature to be used for calling third party function service REST API.

customExternalMappingFunction is the function name that needs to be invoked by third party function service.

TRANSFORM is a keyword that indicates that function is called on the context of transformation and respective API will get called.

params is an optional parameter. There can be infinite optional parameters.

Testing Launch

Once the file is uploaded and all configurations are complete, along with the third party function service up and running, you can proceed with the publish and migration processes. The expected payload can then be verified accordingly.

Troubleshoot Integration Errors

Some of the integration errors and the troubleshooting tips are mentioned below.

Table 7-3 Integration Errors

Error Type	Error Details	Troubleshooting Tips
API configuration	"No corresponding routing solution found for: apiIntegration/v1/preTransform or Transform"	Check TIC configurations are correctly set.
Authentication	401 Unauthorized	Verify credentials and token expiration for External Function Service access.
Function not returning proper result	The supplied @invokeExtFunction response JSON is not of valid syntax.	Validate the return values using API definition mentioned in Appendix.

Table 7-3 (Cont.) Integration Errors

Error Type	Error Details	Troubleshooting Tips
Network	A connection reset error	Check for any network restrictions or proxy settings that might interfere. If the service implemented is deployed on a cloud platform or on premise, make sure it is accessible over the web and cater to the proxy configurations.
Exception in logs	The transformation function @invokeExtFunction must contain at least one argument which specifies the API Path to be invoked as part of the REST URL.	Correct signature of invokeExtFunction as per documentation.

8

Detailed Implementation Steps

Configuring Fabric

For Launch to talk to Routing microservice (RMS), and the outbound call from RMS to Third Party CMS client to happen, there needs to be some configurations done of the CXIF fabric cluster to mediate the flow.

This configuration is to be done using the CXIF Admin API's and more details on the individual service can be found in the Related Guides section.

Create a New Connection Descriptor (TIC)

Connection Descriptor or TIC is where one would mention the host of the third-party CMS client deployed. Also, the authentication mode and the authentication secrets are to be passed in the request while creating TIC.

Supported authentication modes and sample requests for the same are provided below for reference.

> **Note**
>
> Do not modify the fields **endpoint-name**, **type**, and **system-descriptor**. You can modify other fields as per your implementation.

Sample Request:

```
POST https://{CXIFHost}/admin/connectionDescriptors
{
  "endpoint-name": "thirdpartycms",
  "endpoint-url": "https://your-host-example.com/",
  "fabric-facing-auth": {
    "oidc-client-credentials": {
      "client-id": "your-client-id",
      "client-secret": "your-client-secret",
      "identity-uri": "https://your-identity-provider.com/oauth2/v1/token",
      "scope": "your-scope"
    }
  },
  "type": "external",
  "system-descriptor": "thirdpartycms-ttd"
}
```

endpoint-url field should be the host of the service which interacts with the underlying CMS.

fabric-facing-auth field should be where we decide the mode of authentication and the credentials for authenticating.

Replace placeholders with your actual CMS endpoint and OAuth2 credentials. Currently fabric supports three types of authentications. BasicAuth, OAuth2 (only client credentials is supported) and OCIHttpSignature.

The sample payload for each type is as mentioned below.

Table 8-1 Sample Payload

Authentication Type	Sample Payload

Basic Auth	<pre>{
	"system-descriptor": "thirdpartycms-ttd",
	"endpoint-name": "thirdpartycms",
	"endpoint-url": "https://thirdpartycms.dev.com/",
	"fabric-facing-auth":{
	"basic":{
	"username": "admin",
	"password": "*password*"
	}
	},
	"type": "external"
	}</pre>

OAuth2	<pre>{
	"endpoint-name": "thirdpartycms",
	"endpoint-url": "https://thirdpartycms.dev.com/",
	"fabric-facing-auth": {
	"oidc-client-credentials": {
	"client-id":"48eb39a9a7cb4bc0b7761ebb8d3ada97",
	"client-secret":"adt2cdde-6c94-4be2-b525-
	fff575a9c3fc",
	"identity-uri": "https://
	idcs-322c58839e042ad2.identity.oraclecloud.com/oauth2/v1/
	token",
	"scope": "https://n6jfpge6uqfrum.apigateway.us-
	ashburn-1.oci.customer-oci.comurn:opc:resource:consumer::"
	}
	},
	"type": "external",
	"system-descriptor": "thirdpartycms-ttd"
	}</pre>

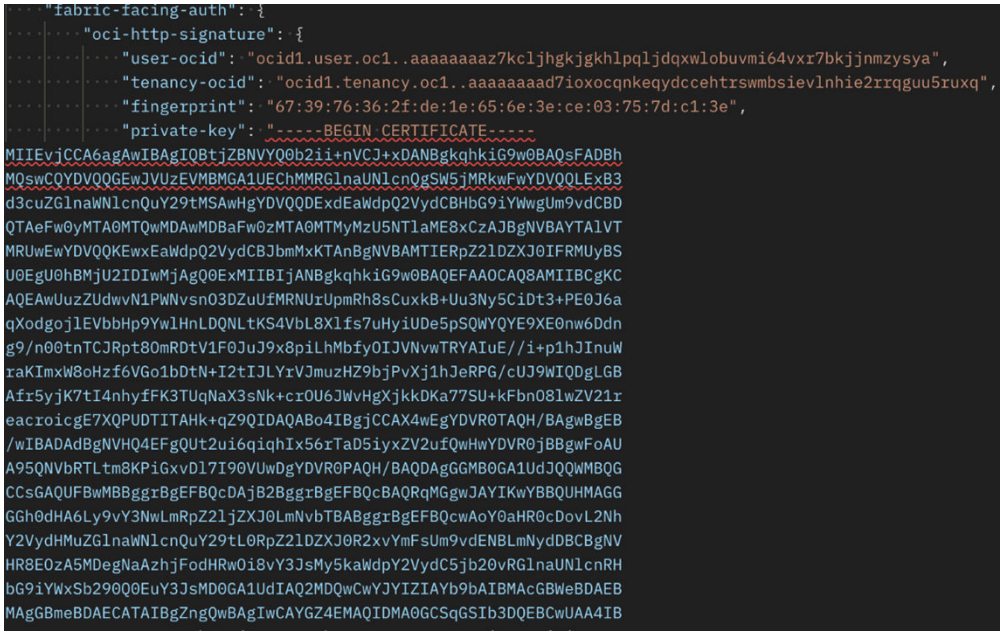
Table 8-1 (Cont.) Sample Payload

Authentication Type	Sample Payload

OCIHttpSignature	<pre>{
	  "system-descriptor": "thirdpartycms-ttd",
	  "endpoint-name": "thirdpartycms",
	  "endpoint-url": "https://thirdpartycms.dev.com/",
	  "fabric-facing-auth": {
	  "oci-http-signature": {
	  "user-ocid":
	"ocid1.user.oc1..aaaaaaaaz7kcljhgkjgkhlpqljdqxwlobuvmi64vxr7b
	kjjnmzysya",
	  "tenancy-ocid":
	"ocid1.tenancy.oc1..aaaaaaaad7ioxocqnkeqydccehtrswmbsievlnhie
	2rrqguu5ruxq",
	  "fingerprint":
	"67:39:76:36:2f:de:1e:65:6e:3e:ce:03:75:7d:c1:3e",
	  "private-key": "-----BEGIN PRIVATE KEY-----\
	\n+pIkanDm==\\n-----END PRIVATE KEY-----",
	  "algorithm": "SHA256withRSA"
	  }
	  },
	  "type": "external"
	}</pre>

> **Note**
>
> In the case of OCIHttpSignature, the postman or other similar tools may not accept the value of “private-key” with multiline (will error out as shown in Figure 8-1).

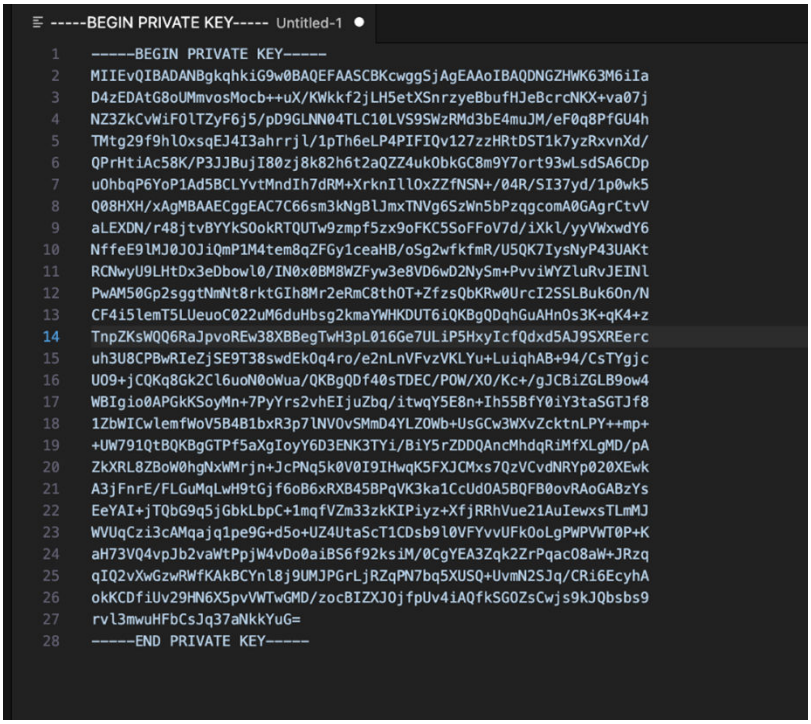
Figure 8-1 Error for OCIHttpSignature



To get around this, format the private key as follows:

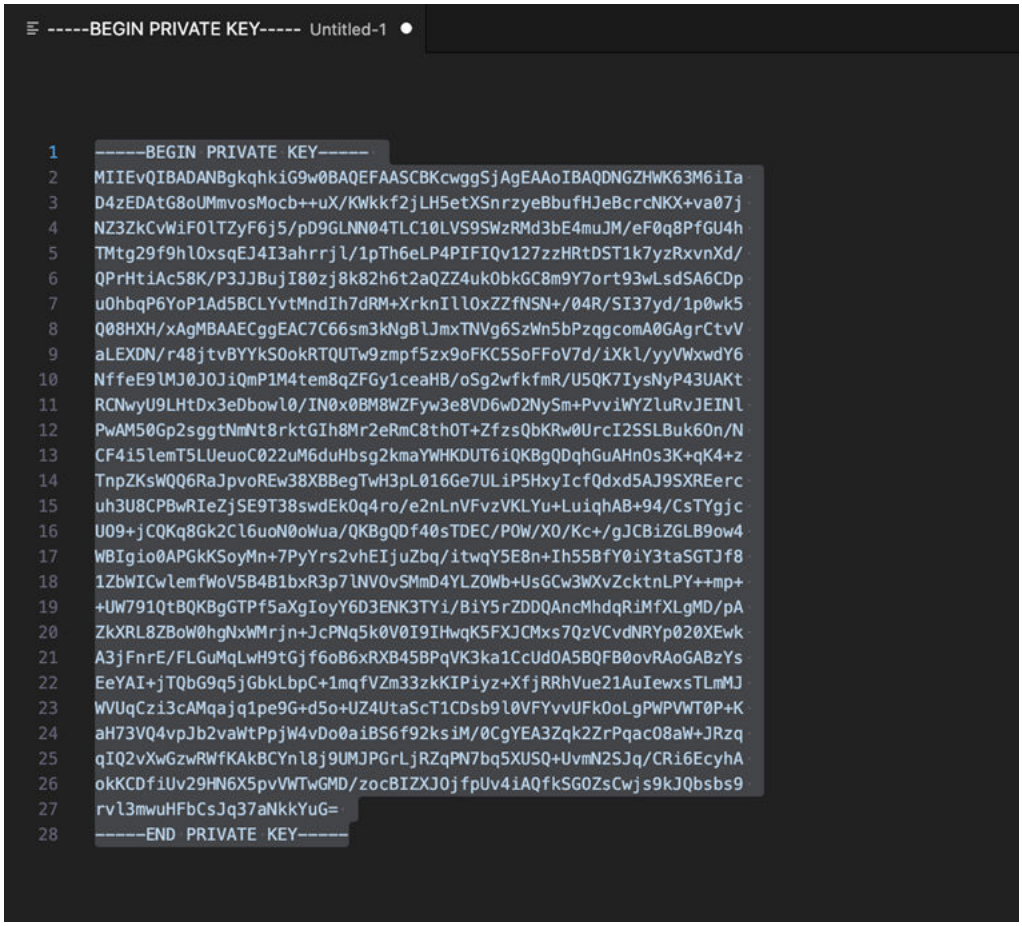
- Open the text editor and paste the private key.

Figure 8-2 Step for Formatting Private Key



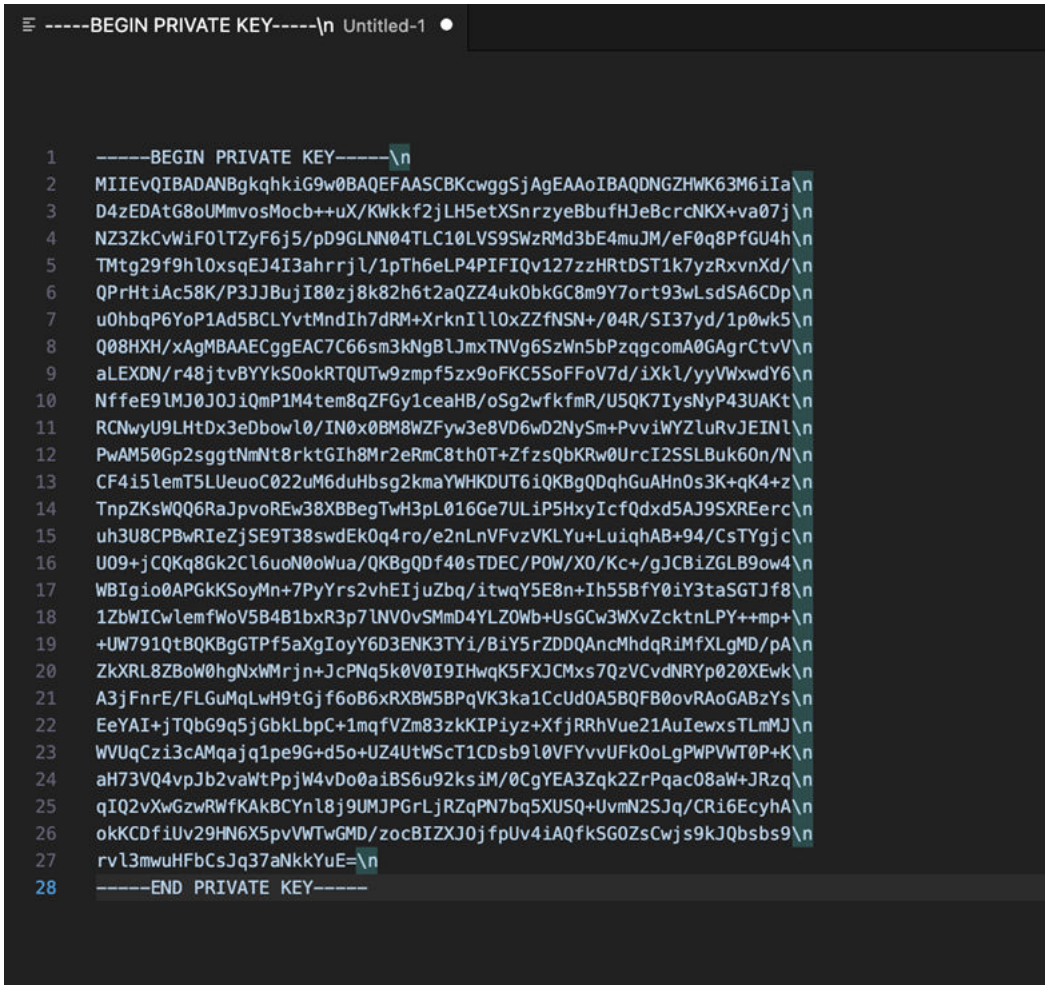
- All the line ends would have an additional line break empty character (see Figure 8-3).

Figure 8-3 Private Key with Additional Line Break Empty Character



- Append “\n” to the end of each line (see Figure 8-4).

Figure 8-4 Private Key Appended with \n

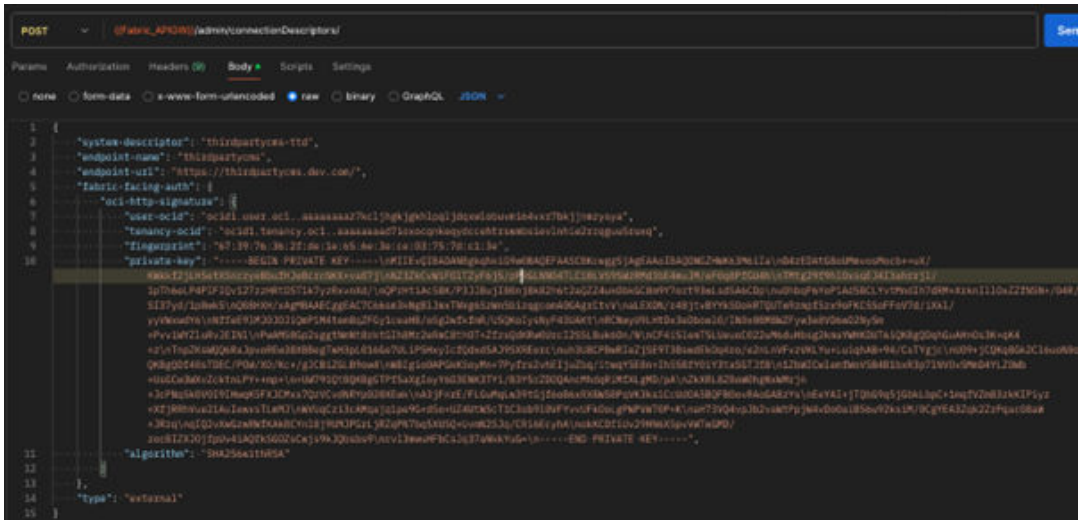


- Now make the entire string single line.

Figure 8-5 Private Key in Single Line

- Now use the same value in request.

Figure 8-6 Private Key (in Single Line) in the Payload



Update Gatekeeping Rules

After configuring TIC or connection descriptor, you need to update the gatekeeping rule which will be auto generated after TIC is generated successfully.

Prepare a GET to

```
https://{CXIFHost}/admin/gatekeepingRules
```

In the response, search for “thirdpartycms” and get the corresponding ID.

Prepare a GET to

```
https://{CXIFHost}/admin/gatekeepingRules/{id}
```

Along with the response of the above GET response, append the below JSON attribute highlighted in red to create full payload and call PUT endpoint.

Prepare PUT to

```
https://{CXIFHost}/admin/gatekeepingRules/{id}
```

```
{
    "endpoint-name": "thirdpartycms",
    "rule-name": "Generated gatekeeping rule for endpoint thirdpartycms",
    "id": "gkr-thirdpartycmslfgstr",
    "destination-selection": [
        {
            "api-id": "extcms-100",
            "api-version": "v1",
            "criteria": [
                {
                    "rank": 10,
                    "resource-ids": [
                        "thirdpartycms"
                    ]
                }
            ]
        }
    ] }
```

Validating the Connection

After configuring Fabric, it's crucial to test the connection before proceeding to Launch configuration.

Prepare a POST request to

```
https://<fabricHost>/api/api/cms/v1/data?limit=3&offset=0
```

with payload.

Sample Payload

```
{
  "filters": [
    {
      "filterName": "attachmentType",
      "filterValue": "images"
    }
  ],
  "additionalParams": [
    {
      "paramName": "environment",
      "paramValue": "prod"
    },
    {
      "paramName": "workspace",
      "paramValue": "telco"
    }
  ],
  "sort": "field_name",
  "order": "asc"
}
```

Send the request and analyze the response. You should receive a JSON object containing CMS data and pagination information.

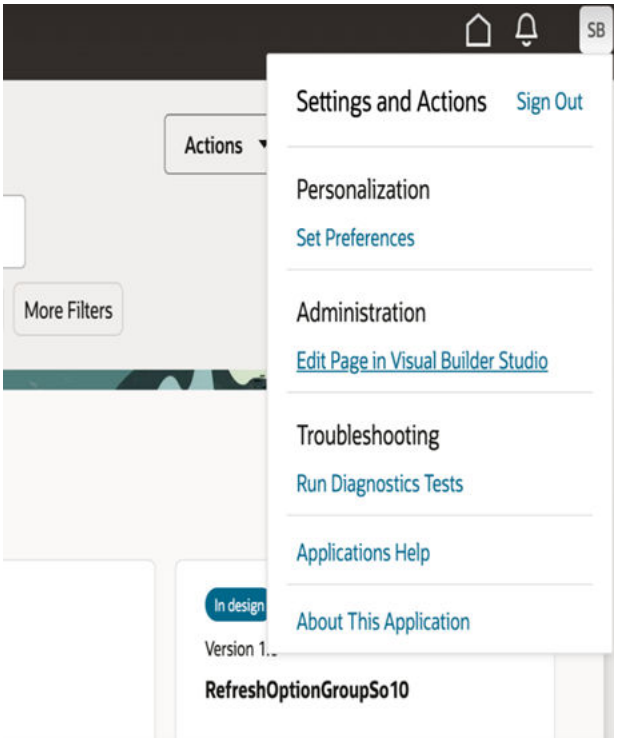
Configuring Launch

Enable Third Party CMS in Visual Builder Studio

To enable third party CMS in Visual Builder Studio:

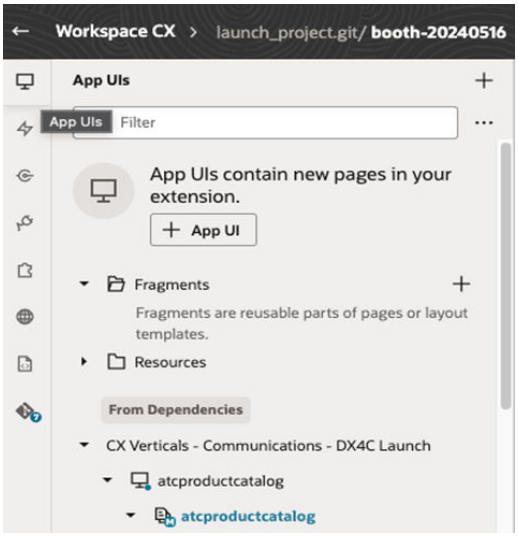
1. Log in to the Launch UI as a user with admin role.
2. In the Launch UI, click your profile icon.
3. Click **Edit Page In Visual Builder Studio** to open Visual Builder Studio.

Figure 8-7 Profile Menu in Launch UI



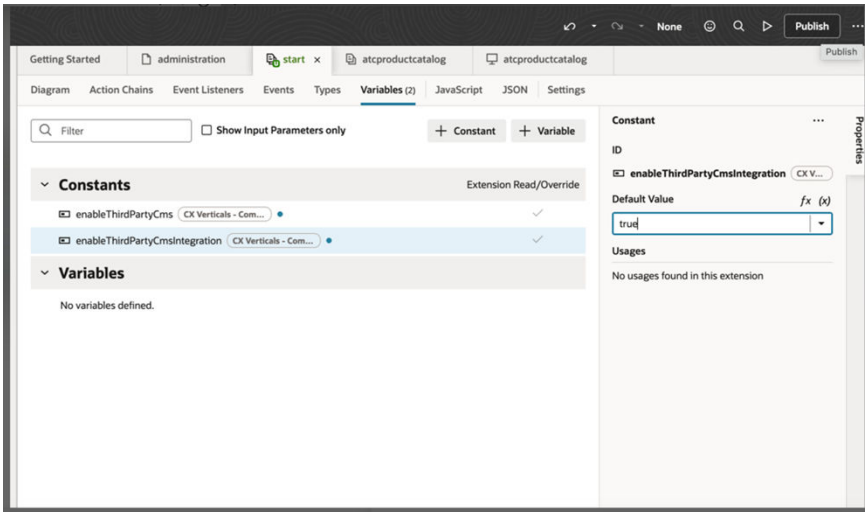
4. In the Visual Builder Studio, ensure that you are on the **App UIs** section by clicking the monitor icon as shown in Figure 8-8.

Figure 8-8 Visual Builder Studio



5. Click the second **atcproductcatalog**.
6. Find the file containing the **enableThirdPartyCmsIntegration** variable (usually a configuration or constants file).
7. Change the value of **enableThirdPartyCmsIntegration** to **true**.

Figure 8-9 Configuring enableThirdPartyCmsIntegration



8. Click **Publish** to save and apply the changes.

Configure Additional Parameters (Optional)

Additional parameters are static part of the request that is always in the payload. If the concrete implementation requires some details always as part of the request, those can be configured here.

If your CMS implementation requires additional parameters, use the attachment endpoint in Launch to upload the additional parameters as a valid JSON array.

Sample Request in CURL format

```
curl --location --request PUT 'https://{AppsHost}/crmRestApi/atcProductCatalog/
11.13.18.05/v1/attachment/' \
--header 'Authorization: Basic <your-base64-encoded-credentials>' \
--form 'primaryFile=@"/path/to/your/parameters.json"' \
--form 'path="thirdPartyCMSParamters"'
```

Sample Payload

```
[
    {
      "paramName": "environment",
      "paramValue": "staging"
    },
    {
      "paramName": "cmsSystem",
      "paramValue": "Contentful"
    }
  ]
```

In the sample, you can see two parameters: **staging** and **cmsSystem**.

If the swagger client implementation has code abstractions or branching flows to connect to different systems or environments of the underlying CMS or any other scenarios where some values need to be part of the request, then it can be configured as mentioned in the steps. The values of both **paramName** and **paramValue** can be free text. It depends on the CMS client implementation.

More examples of requests and responses can be found in swagger.

9
Testing the Integration

To test the integration:

1. Log in to LaunchX.
2. Navigate to any offer or content page that should display CMS content.
3. Click **Attach Image** button or equivalent.
4. In the opened drawer, verify that:
   - You can see filter options specific to your CMS.
   - You can search and browse content from the CMS.
   - You can select and attach content successfully.
5. After attaching content, verify that it displays correctly on the page.

10

Troubleshooting

If you encounter issues:

1. API Configuration: Double-check all API endpoints and IDs in Fabric configuration.
2. Authentication: Verify credentials and token expiration for CMS access.
3. Launch Configuration: Confirm **enableThirdPartyCmsIntegration** is set to **true**. See Enable Third Party CMS in Visual Builder Studio.
4. Network: Check for any network restrictions or proxy settings that might interfere. If the service implemented is deployed on a cloud platform or on premise, make sure it is accessible over the web and cater to the proxy configurations.
5. Logging: Enable debug logging in Fabric and Launch for debugging. You can raise an SRE Ops ticket to change the log level and get the logs.

   Launch logs would only be needed rarely as you can get sufficient details about the request from network tab itself.

   If everything from Launch side looks good, then the next logical step is to get the logs of RMS aka Routing MS. By default, the log level will be error. If the error or issue is not clear, get the log levels raised to DEBUG and then collect the logs. All this log collection and log level changing can be done via raising a ticket to the SRE Ops.

A
Appendix

Setting Default Entities

This appendix provides details and sample payloads for the default entities that must be configured in Launch to ensure smooth integration and migration from Siebel CRM.

Service Specification

Entity Name: Siebel-Default-SS

Description: It is mandatory to have a Service Specification in Launch to create Service offers.

Sample Payload:

```
{
    "id": "Siebel-Default-SS",
    "name": "Siebel Default SS",
    "description": "Default Service Spec for Siebel Product Spec",
    "version": "1.0",
    "lifecycleStatus": "Active",
    "isBundle": false,
    "@type": "ServiceSpecificationOracle",
    "validFor": {
        "startDateTime": "2010-06-19T16:42:23.000Z"
    }
}
```

> **Note**
>
> - When creating a Service Specification, ensure the API request includes the query parameter `skipFeatureProfile=true` in the endpoint URL.
>   Sample API Call:
>   - **Method**: PUT
>   - **Endpoint**: `{{FA_APIGW}}/api/serviceCatalogManagement/v3/serviceSpecification/Siebel-Default-SS?skipFeatureProfile=true`

Product Specification

Entity Name: Siebel-Default-PS

Description: Siebel CRM allows to create simple products without association with classes. However, it is mandatory to specify a product specification in Launch for device and service offers.

Sample Payload:

```
 {
    "id": "Siebel-Default-PS",
    "name": "Siebel Default PS",
    "description": "Default Product Spec for Siebel
Migration",
    "version": "1.0",
    "lifecycleStatus": "Active",
    "isBundle": false,
    "@type": "ProductSpecificationOracle",
    "validFor": {
        "startDateTime": "2010-06-19T16:42:23.000Z"
    },
    "serviceSpecification": [
        {
            "id": "Siebel-Default-SS",
            "name": "Siebel Default SS",
            "version": "1.0",
            "@type": "ServiceSpecificationOracle",
            "@referredType":
"ServiceSpecificationOracle",
            "role": "PRIMARY"
        }
    ]
 }
```

Price List

Entity Name: Siebel-Default-PL

```
 {
    "id": "Siebel-Default-PL",
    "name": "Siebel Default PL",
    "version": "1.0",
    "lifecycleStatus": "Active",
    "@type": "PricelistOracle",
    "validFor": {
        "startDateTime": "2021-09-09T17:16:49.496Z"
    },
    "balanceElement": {
        "id": "USACurrency",
        "name": "USA Currency",
        "@type": "BalanceElementRef",
        "@referredType": "BalanceElementOracle"
    },
    "pricelistType": "RESIDENTIAL",
    "currency": "USD"
 }
```

Custom Profile Specification (Aggregate Discount)

Entity Name: Siebel Default Aggregate Discount CPS

Description: The migration process is expected to have a custom profile specification with quantity and product offer to be available for a successful formation of aggregate discounts.

Sample Payload:


```
 {
    "id": "Siebel-Default-AggDisc-CPS",
    "name": "Siebel Default Aggregate Discount CPS",
    "description": "Default Custom Profile Spec",
    "version": "1.0",
    "lifecycleStatus": "Active",
    "profileType": "DEVICE_SPEC",
    "@type": "CustomProfileSpecificationOracle",
    "validFor": {
        "startDateTime": "2022-02-19T16:42:23.000Z"
    },
    "customProfileSpecChar": [
        {
            "name": "Product Offering",
            "valueType": "PRODUCT_OFFER",
            "@type": "ProductOfferingOracle",
            "configurable": true,
            "minCardinality": 0,
            "maxCardinality": 1,
            "validFor": {
                "startDateTime":
"2022-02-22T00:00:00.000Z"
            }
        },
        {
            "name": "Quantity",
            "valueType": "NUMBER",
            "@type": "CustomProfileSpecChar",
            "configurable": true,
            "minCardinality": 0,
            "maxCardinality": 1,
            "validFor": {
                "startDateTime":
"2022-02-22T00:00:00.000Z"
            }
        }
    ]
 }
```


Custom Profile Specification (Discount Matrix)

Entity Name: Siebel Discount Matrix CPS

Description: The migration process is expected to have a custom profile specification with values present in Siebel.

Sample Payload:


```
{
    "id": "Siebel-Default-DiscMat-CPS",
    "name": "Siebel Default Discount Matrice CPS",
    "description": "Default Custom Profile Spec",
    "version": "1.0",
    "lifecycleStatus": "Active",
```

```
    "created": "2022-07-05T11:07:36.000Z",
    "createdBy": "booth",
    "lastUpdate": "2022-07-19T09:33:13.499Z",
    "lastUpdatedBy": "booth",
    "@type": "CustomProfileSpecificationOracle",
    "validFor": {
        "startDateTime": "2022-02-19T16:42:23.000Z"
    },
    "profileType": "DEVICE_SPEC",
    "customProfileSpecChar": [
        {
            "name": "Account Type",
            "valueType": "STRING",
            "@type": "CustomProfileSpecChar",
            "configurable": true,
            "minCardinality": 0,
            "maxCardinality": 1,
            "validFor": {
                "startDateTime":
"2022-02-22T00:00:00.000Z"
            },
            "customProfileSpecCharValue": [
                {
                    "value": "Clinic",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Commercial",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Company",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Competing Dealer",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Competing OEM",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
```

```
                    },
                    {
                        "value": "Competitor",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Consultant",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Contact Us",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Contract Manufacturer",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Convention Center",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Corporate Training Center",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Corporate/Transient",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Corporation",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
```

```
                    },
                    {
                        "value": "Dealer",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Department",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Department Group",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Military",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "QSR",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Ship To",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Convenience Store",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Manufacturer Rep",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
```

```
                    },
                    {
                        "value": "ODM",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Design House",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "3rd Party Training Center",
                        "valueType": "STRING",
                        "@type":
 "StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "All Suite",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Auto/Home Supply Store",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Banking",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Body Shop",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Branch",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
```

```
                    },
                    {
                        "value": "Broker",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Business",
                        "valueType": "STRING",
                        "@type":
 "StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Business Customer",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Advertiser",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Central Bank",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Chain Drug",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Chain Food",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Department Store",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
```

```
                    },
                    {
                        "value": "Residential",
                        "valueType": "STRING",
                        "@type":
 "StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Customer",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Agency",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Vendor",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Committee",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Contract Research
Organization",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Surgery",
                        "valueType": "STRING",
                        "@type":
"StringCharacteristicValueSpecification",
                        "isDefault": false
                    },
                    {
                        "value": "Chemist",
                        "valueType": "STRING",
"StringCharacteristicValueSpecification",
                        "isDefault": false
```

```
                },
                {
                    "value": "Clinical Directorate",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "District Health
Authority",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Drug Committee",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Hospital Unit",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Practice",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "PBM",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                },
                {
                    "value": "Pharmaceutical Company",
                    "valueType": "STRING",
                    "@type":
"StringCharacteristicValueSpecification",
                    "isDefault": false
                }
            ]
        },
        {
            "name": "Product Offering",
            "valueType": "PRODUCT_OFFER",
```

```
            "@type": "ProductOfferingOracle",
            "configurable": true,
            "minCardinality": 0,
            "maxCardinality": 1,
            "validFor": {
                "startDateTime":
"2022-02-22T00:00:00.000Z"
            }
        }
    ]
}
```

> **Note**
>
> The lifecycle Status for these default objects must be set to **Active**. To do this, send a PATCH request to the respective REST APIs.

Downloading Third Party CMS Swagger

Use this GET request to download the Third Party CMS Swagger.

```
GET - https://{FAHOST}/crmRestApi/atcProductCatalog/11.13.18.05/
productCatalogManagement/v1/swagger/ThirdPartyCMSSwagger
```

This comprehensive API reference will provide in-depth information on all available operations and data structures and different sample requests and responses for the CMS integration.

Downloading Third Party Function Service Swagger

Use this GET request to download the Third-Party Function Service Swagger:

```
GET - https://{FAHOST}/crmRestApi/atcProductCatalog/11.13.18.05/
productCatalogManagement/v1/swagger/PreTransformExternalFunction
```

This comprehensive API reference will provide in-depth information on all available operations and data structures and different sample requests and responses for the Third-Party Function service integration for both Transform and PreTransform API's.