Oracle® Communications Launch Cloud Service

Integration Guide

Release 25A
G12920-01
January 2025

ORACLE

Oracle Communications Launch Cloud Service Integration Guide, Release 25A

G12920-01

Contents

A Appendix

Preface

The Oracle Communications Launch Cloud Service Integration Guide describes the prebuilt integration between Launch and Siebel CRM.

Audience

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Launch Cloud Service Siebel CRM Integration

The prebuilt 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 prebuilt integration between the two applications, you can perform a one-time migration of catalog definitions from Siebel to establish them as the authoritative source in Launch.

Siebel customers can innovate faster by centrally managing their product portfolio for their 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 24.11 Update Guide & Known Issues	Describes the changes in the Siebel REST APIs supporting the integration.

Supported Versions

- Launch version 25.01 or later
- Siebel CRM version 24.11 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.

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)
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
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

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.

Setup Task List

Table 1-3 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.

Table 1-3 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.	Launch	Register destinations	Yes	This is required to configure the right spoke instance to receive the publishing events.
3.	Launch	Configure Entity Profile	Yes	This is required to ensure that Launch can model catalog definitions based on Siebel.
4.	Launch	Setup Default entities	Yes	This is required to meet the Launch requirement of Products and Services to have specifications.
5.	Industry Framework	Configure the spoke systems	Yes	This is required to ensure to configure the spoke system instance for receiving publishing events.
6.	Siebel	Create client credentials and security requirements	Yes	N/A
7.	Siebel	Migrate Literature to Oracle Content Management	No	This is required only if images of devices, sales collaterals are associated to product definitions in Siebel.

Table 1-3 (Cont.) Setup Task List

No.	Application	Task	Mandatory?	Description
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.

Setup Task Details

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. Register **Destination** in Launch for publishing to the Siebel instance. There are multiple Siebel 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 Catalog Sync configuration.
   c. Internal - Set this to ON.

   To create a Siebel destination:

   a. Navigate to Launch user interface > **Administration** > **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 Catalog Entities" 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 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.

3. 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 default specification by default in Launch in case of 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 payloads, see "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 payloads, see "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 Customer Profile Specification characteristics to support Attribute based adjustments. Such definitions are published to Siebel CRM as Discount matrix with Class and Customer attributes. Since Siebel CRM does not support Service Specification based ABP, use product specifications and customer profile specifications to migrate the Discount matrix from Siebel CRM and subsequent publishing to Siebel CRM.

> **Note:**
>
> Ensure that the customer profile in Siebel CRM is synced to Customer Profile Spec before the initial migration. Refer to the "Appendix" for a sample payload.

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. 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.

6. Create client credentials and security requirements for Siebel CRM. See Overview of *Using Siebel REST API* and *Using REST API For Siebel Telco* in the Siebel REST API Guide.

7. Migrate Literature to your 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 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*.

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 data for the 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.

    c. Ensure that the product definitions have an effective start date. This is required in Launch as it drives the validity of associations to other entities.

    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 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.

h. 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 and Publishing Process

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 *Launch Cloud Service Implementation Guide*.

Migration of Siebel entities is one-time to be used for onboarding a Siebel customer process with Launch being baselined 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 using the Publish 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 *Launch Cloud Service Implementation Guide*.

Publish will release all the Siebel versioned objects during the publish process for the below versioned objects:

- Product Class
- Attributes
- Product Definitions
- Promotions

Enrichment After Publishing

To enrich objects in Siebel CRM after they have been published in Launch, you need to lock the versions, 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 these 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 (See "Setup Task Details".)
- Promotion
  - Merge
  - Split
  - User Interface

Migration Job Parameters

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

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 UI. 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.
- **Header Parameters**: Optional header parameters. For more information, see "Migration API".

Migration APIs

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

- Name match
- ID match
- Wild card search using Name

Table 1-4 contains a list of migration API calls with sample query parameters and their values.

Table 1-4 Migration APIs

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 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*'
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

Table 1-5 contains the complete list of supported migration paths.

Table 1-5 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.
/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.

Table 1-5 (Cont.) Supported Migration Paths

Siebel API	What gets migrated
/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

> **Note:**
>
> 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. For more information, see the topics *Understand Entity Mapping* and *Rest APIs* 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.

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.

2

Launch Cloud Service PDC (BRM) Integration

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.
Oracle PDC-BRM Documentation	PDC/BRM 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 25.01 or later and
- 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 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 pricing	Charge Selector	Definition Pricing
Attribute based adjustment	Discount Selector	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.

Setup Task List

Table 2-3 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.	Launch	Register destinations	Yes	This is required to configure the right spoke instance to receive the publishing events.

Table 2-3 (Cont.) Setup Task List

No.	Application	Task	Mandatory?	Description
3.	Launch	Configure Entity Profile	Yes	This is required to ensure that Launch can model catalog definitions based on PDC/BRM.
4.	Industry Framework	Configure the spoke systems	Yes	This is required to ensure to configure the spoke system instance for receiving publishing events.
5.	PDC REST Services Manager	Create client credentials and security requirements	Yes	N/A
6.	Launch and PDC/BRM	Set up Configuration entities	Yes	This is required to meet the Launch and PDC configuration setup.

Setup Task Details

1. Create a new 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. 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 instance is a destination that needs to be setup for Launch to publish to the correct PDC 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 Catalog Sync configuration.
   c. Internal – Set this to ON.

   Here's the list of steps to create a Siebel destination:

   a. Navigate to the Launch user interface > **Administration** > **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 PDC destination name in the Destinations field and save it.

f. Choose the **Activate** option for the PENDING lifecycle 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 Catalog Entities" 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 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.

3. 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:

   - Both the applications have the same construct capabilities and restrictions for an entity at par - for example, Balance Element, Product Offering, Service Specification entities, and so on.
   - Both the applications have minimal common capabilities and/or additional capabilities or restrictions in one and not in the other - for example, PDC/BRM 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 PDC/BRM package.

4. 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.

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, and balance elements required for charge or discount offers are set appropriately.

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

   a. GLID name on Launch and BRM system should be same.

i. In Launch → Setup and Maintenance, go to standard lookup ORA_ATC_GLID and set the GLID. **Example**: Create 101 GLID in launch.

ii. In PDC, use pin_glid application to create the general ledger ID. Create 101 GLID in PDC.

b. Tax code name on Launch and BRM system should be same.

i. a. In Launch, use the REST API operation to create https:///<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/productCatalogReferenceManagement/v1/taxServiceProvider. **Example**: Tax001 should be setup as taxCode in taxSupplier in Launch.

ii. In PDC, set up the same set of tax codes as configured in Launch.

c. Balance element name, code, numeric code on Launch needs to be same as PDC.

i. In Launch, use the REST API to create the Balance element https:///<HOST>/crmRestApi/atcProductCatalog/11.13.18.05/productCatalogReferenceManagement/v1/balanceElement.**Example**: Name: US Dollar, code - USD, numeric code – 840 and so on.

ii. In PDC, configure the Balance Element (Name, Code, Numeric code), along with other attributes that should match with attributes in Launch.

d. Usage specification name and metering rule name in Launch needs to be same as the configured Usage event and RUM name on PDC side.

i. In Launch, use the REST API to create the usage specification https://<HOST>/crmRestApu/atcProductCatalog/11.13.18.05/tmf-api/usageManagement/v2/usageSpecification. The fields name, meteringRule.name should match with PDC. **Example**: EventDelayedTelcoGSMTelephony, meteringRule.name – Duration

ii. In PDC, configure the serviceEventMap with the same event name and RUM.

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.

Sample Mapping

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

Table 2-4 Sample Mapping

Entity	Launch	PDC
Usage Specification	EventDelayedTelcoGSMTelephony Metering Rule: Duration	EventDelayedTelcoGSMTelephony (Event) RUM: Duration
UOM (Unit of Measure)	ORA_ATC_UOM	UOM
Tax Code	TAX001	TAX001

Table 2-4 (Cont.) Sample Mapping

Entity	Launch	PDC
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.
Product Specification	Wireless PS • Populate the Usage Specification - EventDelayedTelcoGSMTelephony Populate the Service Specification - Wireless CFS	N/A
GLID	101	101
Balance Element	Name: US Dollars Code: USD Numeric code: 840	Name: US Dollars Code: USD Numeric code: 840
Price Tag	CT01	CT01
Impact Category	Common business configuration impact category IC_INTERNATIONAL	IC_INTERNATIONAL

Supported Scenarios

Table 2-5 lists the supported integration scenarios.

Table 2-5 Supported Scenarios

S.No.	What you can publish?	Launch Entity	PDC Entity
1.	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)

Table 2-5 (Cont.) Supported Scenarios

S.No.	What you can publish?	Launch Entity	PDC Entity
2.	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)
3.	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
4.	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
5.	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
6.	Simple Offer - Tiered pricing	Simple offer with one time/recurring tiered pricing	Charge offer of service type with one time/ recurring tiered pricing
7.	Simple offer - Usage tiered pricing	Simple offer with usage tiered pricing	Charge offer with usage tiered pricing

Table 2-5 (Cont.) Supported Scenarios

S.No.	What you can publish?	Launch Entity	PDC Entity
8.	Re-use of price plans in Simple offer	Simple offer with reused price plans	Charge offer with reused rate plans
9.	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
10.	Factor the following integration scenarios cloning, revisions	Clone Simple offer revision Simple offer Retire Simple offer Obsolete	Charge offer revise Charge offer obsolete Charge offer obsolete
11.	Launch - BRM - 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
12.	Launch - BRM - 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
13.	Launch - BRM - 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
14.	Launch BRM - 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
15.	Launch - BRM - 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

Table 2-5 (Cont.) Supported Scenarios

S.No.	What you can publish?	Launch Entity	PDC Entity
16.	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
17.	Service Bundle (No nesting of bundles)	Service bundle	Bundle
18.	Package with commitment terms (no nested bundles or commercial bundles, aggregate groups)	Package	Package with commitment terms
19.	Service Bundle with commitment terms (No nesting of bundles)	Service bundle	Bundle with commitment terms
20.	Package (no nested bundles or commercial bundles, aggregate groups)	Package	Package
21.	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
22.	Publish other type of product offering (like discounts)	Simple offering with other type of product offering + discount	Discount offers
23.	Simple Offer with multiple allowances and consumption model	Simple offer With multiple allowances and consumption model	Charge offer and Discount offer (consumption model)
24.	Simple Offer with usage prices based on zoning	Simple offer with Value Map zoning	Charge offer using charge selector with value map zoning
25.	Simple Offers with adjustments based on triggers	Simple offer, triggers on adjustments (Total Charge, Total Quantity, Price Tag, Expression as trigger conditions)	Discount offer with discount trigger (price tag in Launch maps to impact category in discount filter)
26.	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.
27.	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.

Table 2-5 (Cont.) Supported Scenarios

S.No.	What you can publish?	Launch Entity	PDC Entity
28.	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
29.	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
30.	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
31.	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

3

Launch Cloud Service Third Party Content Management System Integration

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

Table 3-1 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

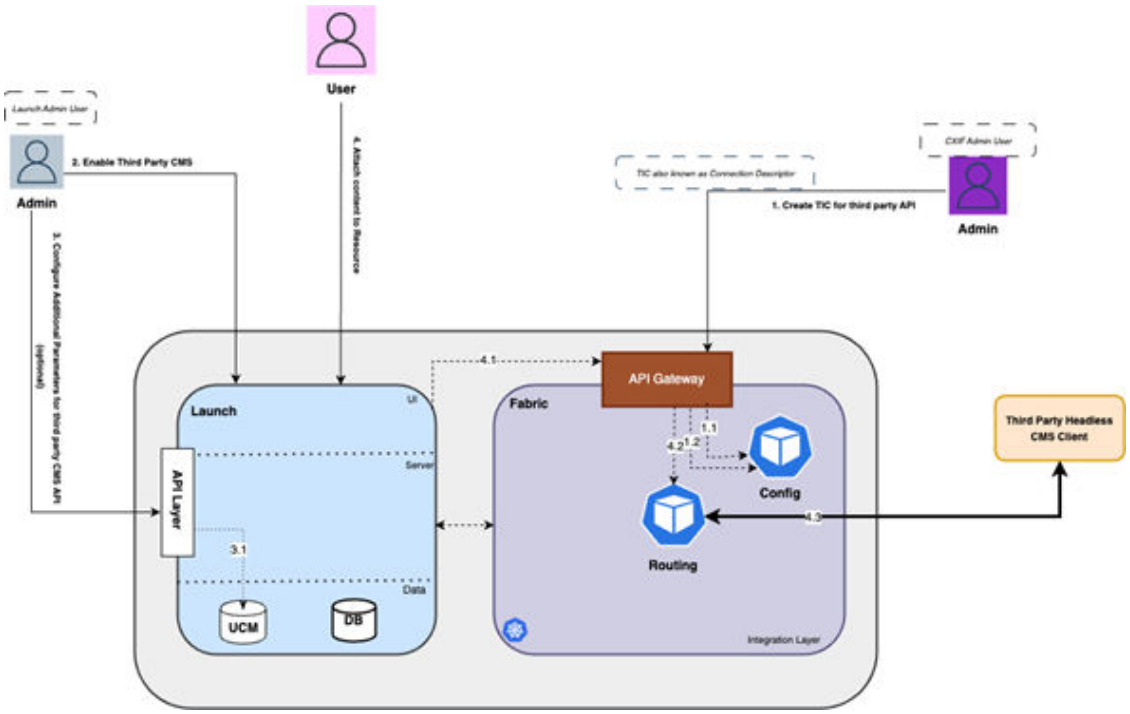
The integration involves three main components:

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 3-1 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 via Visual Builder Studio.
3. If the CMS client needs additional parameters with each request, these are set up as a JSON configuration in Launch 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.

4

Launch Cloud Service External Mapping Services Integration

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 *Integrate Launch with Siebel CRM and Pricing Design Center (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 4-1 contains information about other useful sources of information for the integration process.

Table 4-1 Related Guides for External Mapping Services Integration

Reference	Description
Integrate Launch with Siebel CRM and Pricing Design Center (BRM)	Describes the setup and implementation of the edit and upload mapper file.
Implement CX Industries Framework	Describes the setup and implementation of the CX Industries Framework required for Launch integrations.

System Architecture Overview

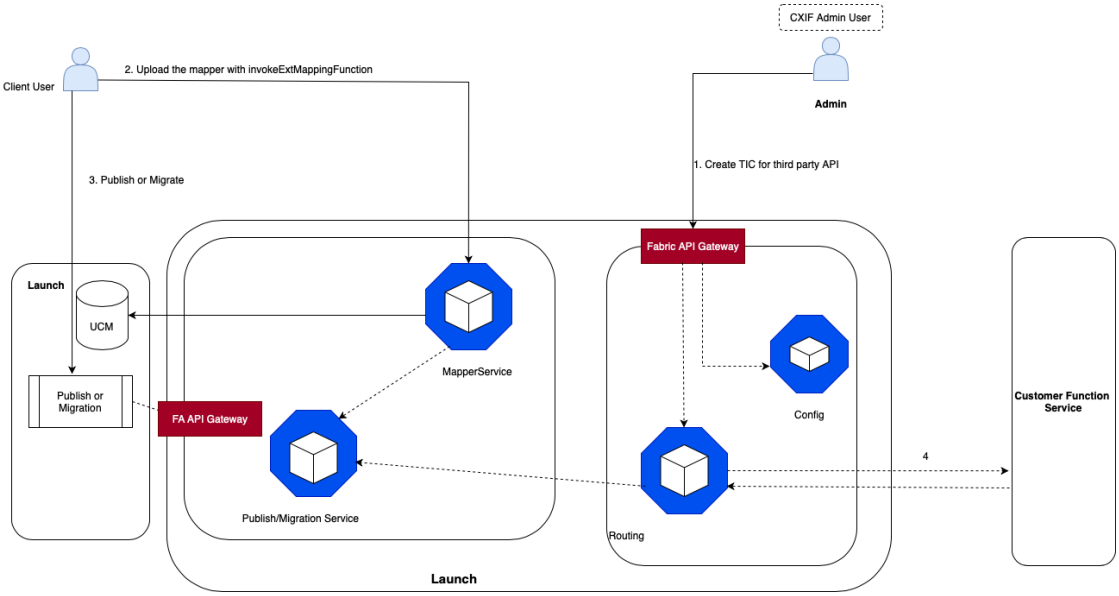
The integration involves three main components:

1. **Fabric**: Acts as the API management and routing layer. Fabric also have Integration microservices for uploading Mapper file and perform publish/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 4-1 Integrating Launch, Fabric, and the third party function service



- **Step 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.
- **Step 2**: The mapper file with the necessary mapping extension changes is uploaded. The file is stored in UCM.
- **Step 3**: The user publishes or migrates the project.
- **Step 4**: The Customer Function Service is called and returns the appropriate response.

Detailed Implementation Steps

Configuring Fabric

For Launch to talk to RMS, and the outbound call from RMS to “Third Party Function Service” to happen, there needs to be some configurations done in the CXIF fabric cluster to mediate the flow. This configuration is to be done using the CXIF Admin API’s. 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 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 4-2 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 4-2 (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.

PreTranform 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": "{}”
}
}
```

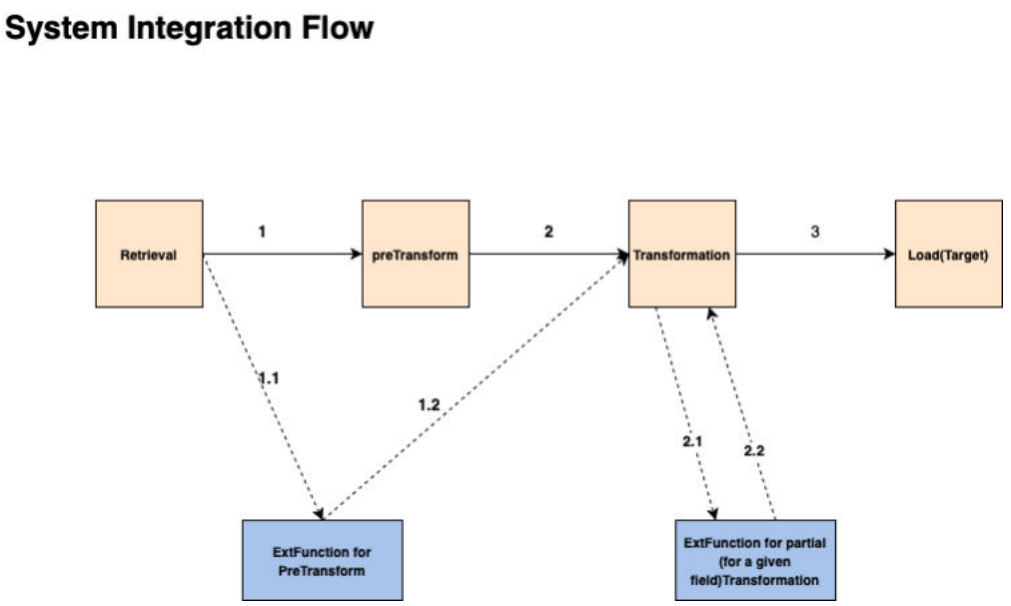
For transform, valueType can be either an object, an array, integer, string, or number and value should be of same data type as mentioned in valueType.

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 4-2 System Integration Flow



In mapper file, source_request_spec defines the procedure of retrieving source data. After the retrieval of source data, pre transformation happens (Figure 4-2 step 1). 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 beginning from root, will be selected 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 (Figure 4-2 step 1.1), the below signature can be used. Once this signature is processed in runtime, the preTranform 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 (Figure 4-2 step 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 (Figure 4-2 step 2.1 and step 2.2).

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 4-3 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 4-3 (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.

5

Detailed Implementation Steps

Configuring Fabric

For Launch to talk to 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 5-1 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 5-1 (Cont.) Sample Payload

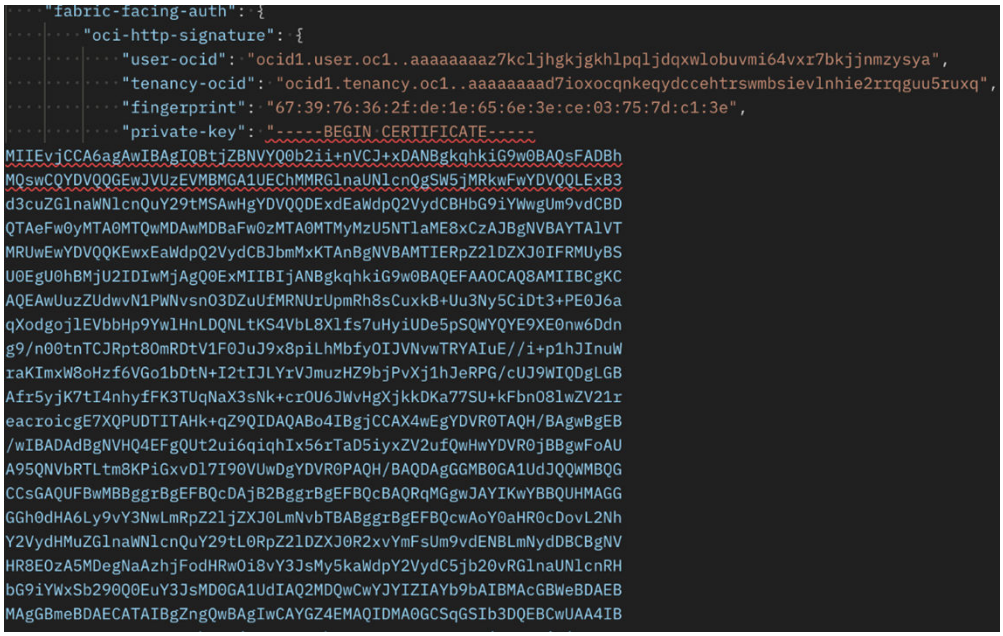
Authentication Type	Sample Payload

OCIHttpSignature	{
	"system-descriptor": "thirdpartycms-ttd",
	"endpoint-name": "thirdpartycms",
	"endpoint-url": "https://thirdpartycms.dev.com/",
	"fabric-facing-auth": {
	"oci-http-signature": {
	"user-ocid":
	"ocid1.user.oc1..aaaaaaaaz7kcljhgkjgkhlpqljdqxwlobuvmi64vxr7bkjjnmzysya",
	"tenancy-ocid":
	"ocid1.tenancy.oc1..aaaaaaaad7ioxocqnkeqydccehtrswmbsievlnhie2rrqguu5ruxq",
	"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"
	}

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 like below image).

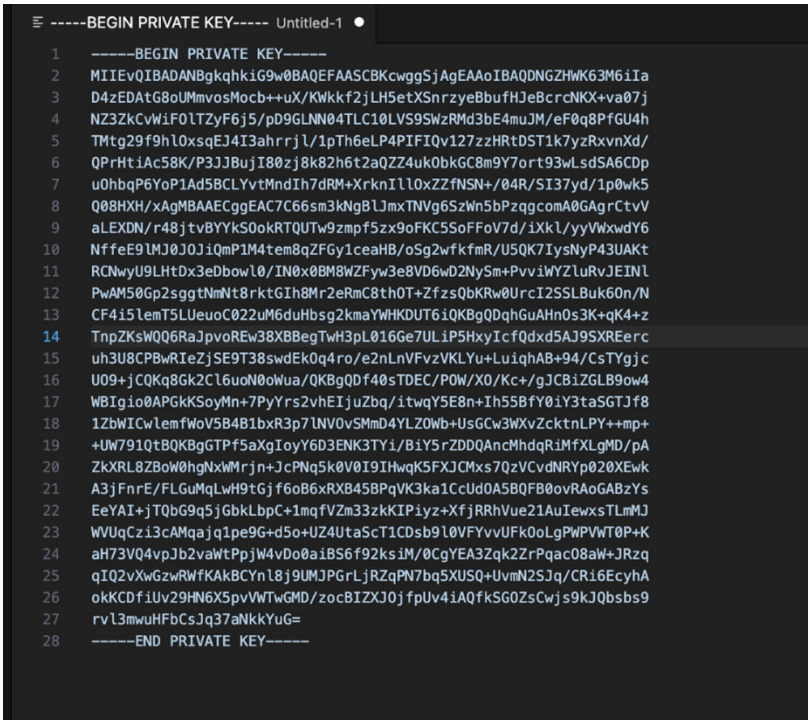
Figure 5-1 Error for OCIHttpSignature



To get around this we need to format the private key as mentioned below.

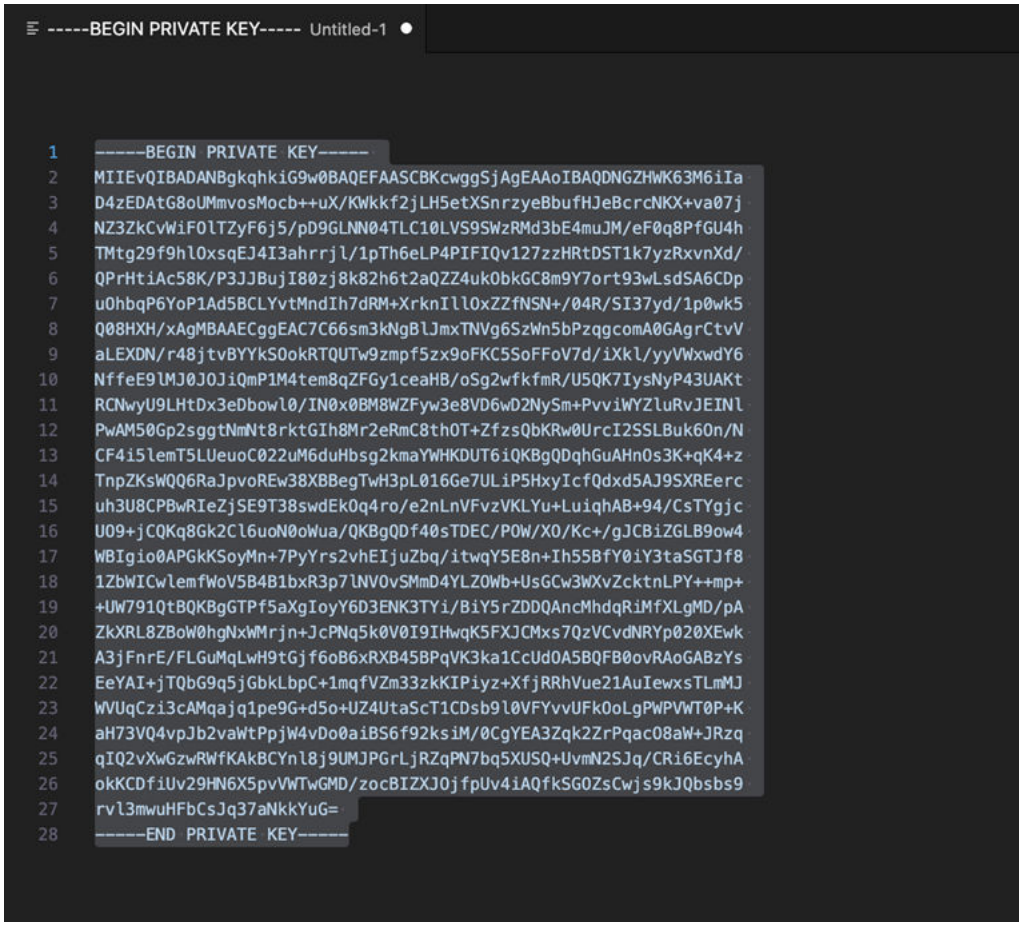
- Open the text editor and paste the private key.

Figure 5-2 Step for Formatting Private Key



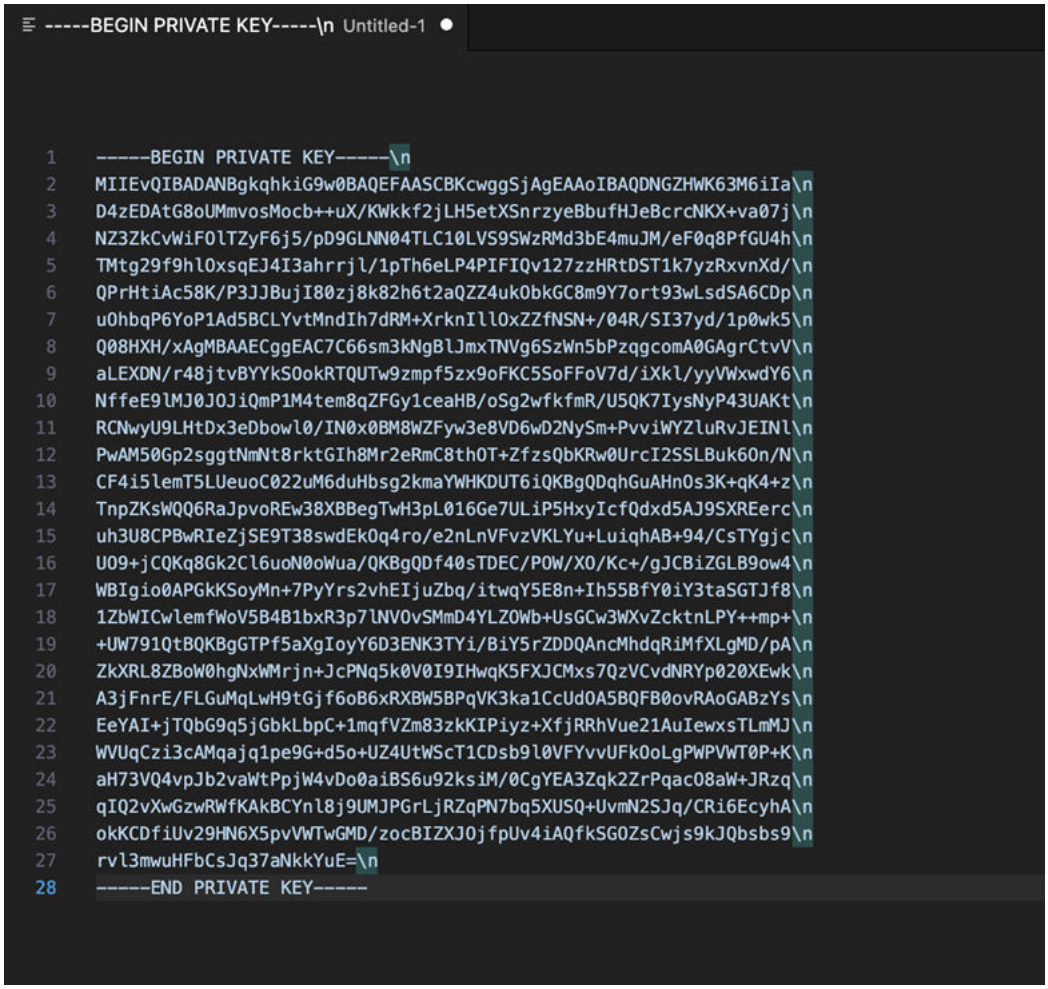
- All the line ends would have an additional line break empty character (see below image).

Figure 5-3 Private Key with additional line break empty character



- Append “\n” to the end of each line (see below image).

Figure 5-4 Private Key with Appended with \n

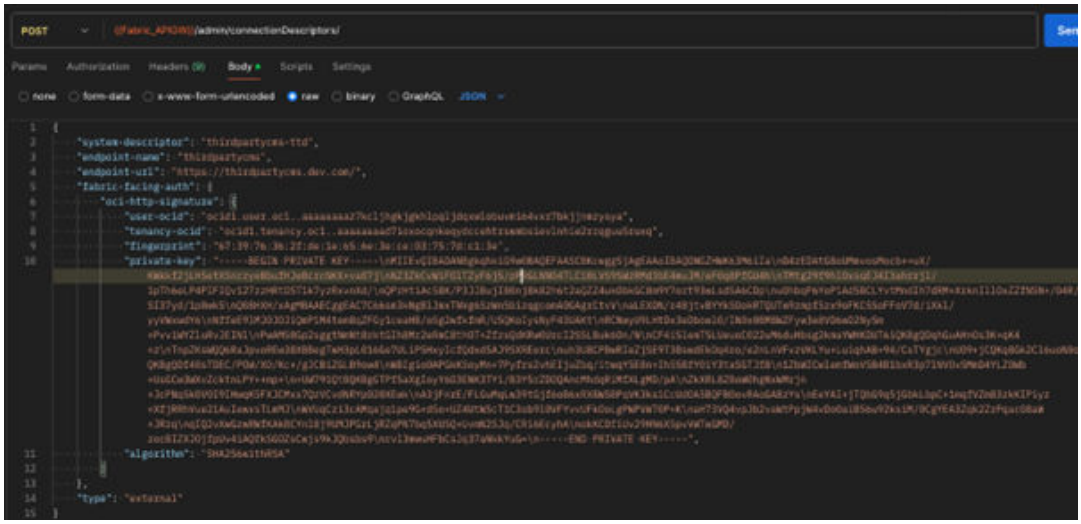


- Now make the entire string single line.

Figure 5-5 String in Single Line

- Now use the same value in request.

Figure 5-6 Same Value in Request



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"
}
```

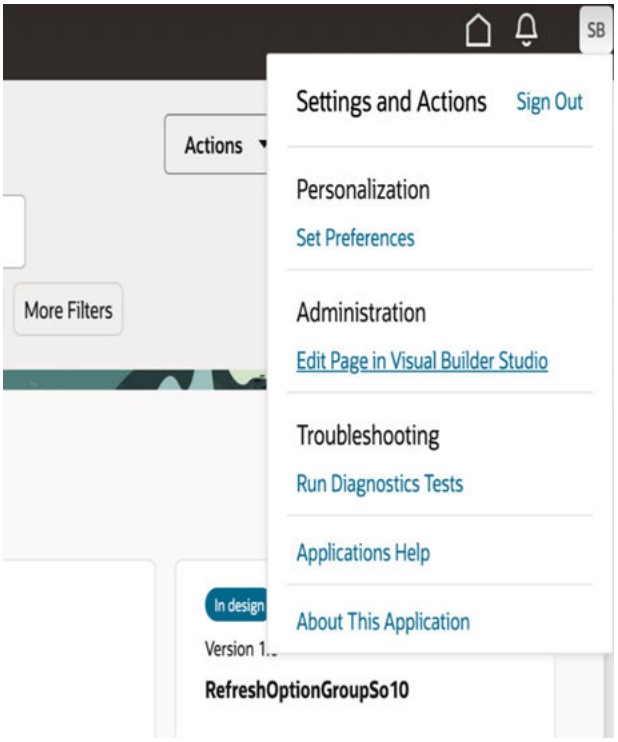
Send the request and analyze the response. You should receive a JSON object containing CMS data and pagination information.

Configuring Launch

Enable Third Party CMS in Visual Builder Studio

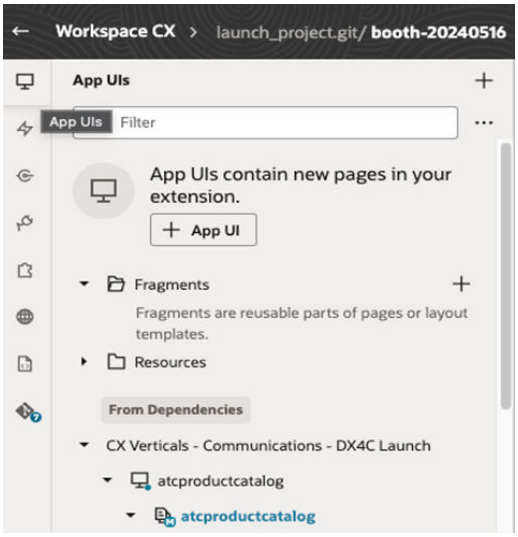
- Login to LaunchX UI as a user with admin role.
- Click the profile icon and then open VB studio by clicking **Edit Page in Visual Builder Studio**.

Figure 5-7 Profile Menu in LaunchX UI



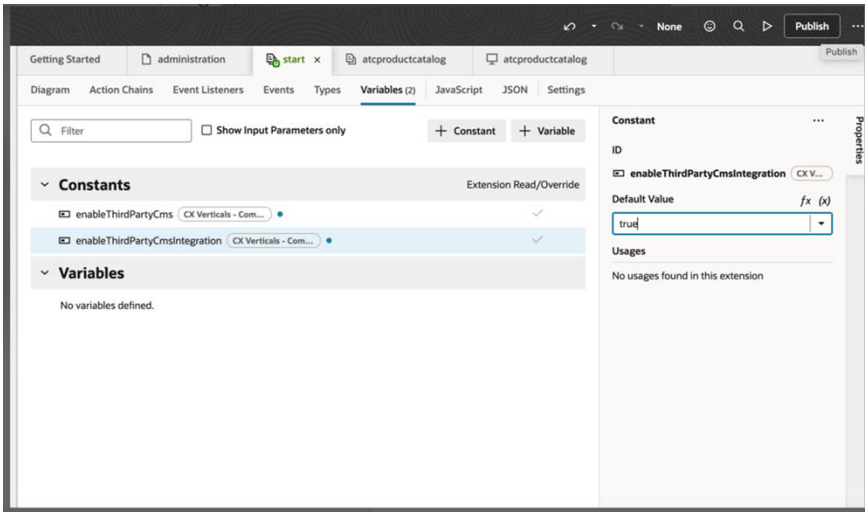
- In the Visual Builder Studio, ensure that you are on the **App UIs** section by clicking the monitor icon as shown in Figure 5-8.

Figure 5-8 Visual Builder Studio



- Click the second **atcproductcatalog**.
- Find the file containing the **enableThirdPartyCmsIntegration** variable (usually a configuration or constants file).
- Change the value of **enableThirdPartyCmsIntegration** to **true**.
- Click **Publish** to save and apply the changes.

Figure 5-9 Publishing Page in Visual Builder Studio



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 like mentioned in 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.

6
Testing 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.

7 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.

8
Source/Target Mapping

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 "Handling Extensions".

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 OAS to understand more on the APIs.

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

Table 8-1 lists the functions available for use in the data transformation, refer to the description column to understand the use of a function.

Table 8-1 Supported Functions

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

Table 8-1 (Cont.) Supported Functions

Function Name	Signature	Description
@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.
@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.

Table 8-1 (Cont.) Supported Functions

Function Name	Signature	Description
@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.
@filterArray	@filterArray(conditionalJsonPath, filteredArrayJsonPath, minLengthForComparing, operator, additionalCondition)	Filter the array based on the condition provided.

Table 8-1 (Cont.) Supported Functions

Function Name	Signature	Description
@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 8-1 (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 8-2 lists the application constants available for use in data transformation, refer to the description column to understand the use of respective application constants.

Table 8-2 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 8-2 (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 number of records when the JSON path returns multiple objects in API map.

@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 8-2 (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 8-3 lists template files that are referenced by the source API configuration during the migration event.

Table 8-3 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 8-3 (Cont.) Templates used by Siebel CRM

Template	Description
ProductPostIdsTemplate.json	Used during nested job processing for Siebel products.

Table 8-4 Templates used by Siebel CRM

Template	Description
catalogIDTTemplate.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.

Managing Mapping File Versions

Oracle ships the two 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.

You can retrieve the mapping file using the following command:

```
GET - {{APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
     Headers to be passed:
         Source System  <<SourceName>>
         X-Destination-System <<DestinationName>>
```

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.

To use versioning of the grammar files you can pass a unique version as part of Mapping APIs using the X-Version header. But it's optional. If the header is not passed, the default version of the mapping is used.

After creating the grammar files using different versions, you can 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, refer to the "Handling Extensions" section below.

Handling Extensions

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 with the required template files for the standard Siebel fields. The seeded mapping can be updated, or new mapping configurations can be created using the REST APIs. This can also be done using the Catalog Sync UI.

The steps to add extension in grammar files:

1. All extensions to Launch for the identified Siebel extensions to product definitions to 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 productOfferingCustomer.yml file.

2. The seeded grammar file can be retrieved by using get Grammar call as below.

a. For migration related changes the X-Source-System should be “siebel” and X-Destination-System should be “launch”.

b. For publishing related changes the X-Source-System should be “launch” and X-Destination-System should be “siebel”.

```
GET - {{APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
     • Headers to be passed:
        o X-source systems <<SourceName>>
        o X-Destination-System <<DestinationName>>
```

3. Once the grammar files are retrieved, the next step is to make the mapping edits:

For example, a new @type is defined as productOfferingCustomer, the changes in target_spec_request will be addition of new target_spec_request

Search for componentName of “SiebelProduct” and copy the entire Json Object as it is into the new file and change the following two fields :

```
"eligibility_check_json_path": "$..projectItems[?(@.isBundle == 'false' && @.
['@referredType'] == 'productOfferingCustomer')]",
"source_data_path": "$..projectItems[?(@.isBundle == 'false' && @.
['@referredType'] == 'productOfferingCustomer')]",
```

Also change the component name and give a new Name:

```
"componentName": " productOfferingCustomer ",
"name": "productOfferingCustomer"
```

Define this newly created target_request_spec just below siebelProduct for convenience.

Define the component created above in component section:

Use all_of of openAPI to refer Siebel product in new component as below:

```
" productOfferingCustomer ": {
 "title": "SiebelProductContainer",
"type": "object",
"allOf": [
{
"$ref": "#/components/schemas/SiebelProduct"
},
{
"properties": {
```

Maintain the tree hierarachy of parent (siebelProduct in the properties, not necessarily all the properties in the tree) and add only the additional extended field.

```
" productOfferingCustomer ": {
"title": "SiebelProductContainer",
"type": "object",
"allOf": [
   {
    "$ref": "#/components/schemas/SiebelProduct"
   },
   {
     "properties": {
      "SyncChild": {
      "type": "string",
        "x-oracle-map-data": {
         "json_path": "$.dummyField",
         "mapType": "TO_FIELD",
         "default": "Y"
   }
  },
  "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": {
   "ExtendedField": {
    "type": "string",
    "x-oracle-map-data": {
     "mapType": "TO_FIELD",
     "json_path": "$.sourceResponse.productOfferingTMF.id"
    }
   },
```

In the above example, a new field named ExtendedField is added inside SWI Product Definition VBC.

Existing fields will be inherited from SiebelProduct , only the tree hierarchy needs to be maintained.

The created grammar file with changes needs to be uploaded to UCM. It is advisable to upload the created grammar file starting with V1 version and then incrementing the version for every subsequent changes , this can be done as below:

```
PUT - {{APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
     • Headers to be passed:
        o X-Source-System <<sourceName>>
        o X-Target-System <<targetName>>
        o X-Version V1
     • Body to be passed
        o File. <<file to be uploaded>>
```

Once the grammar file is uploaded to V1 version , destination needs to updated as below:

```
PATCH -{{launch_URL}}/crmRestApi/resources/11.13.18.05/
atcPublishWorkspaceDestinations
    • Body to be passed
    {
      "DestVersion": "V1"
    }
```

All the template file mentioned in section Templates used by Siebel CRM also needs to be uploaded with X-Version as V1.

```
POST – {{FA_APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/template
      • Headers to be passed
         o X-Source-System- <<SourceName>>
         o X-Target-System- <<targetName>>
         o X-Version-V1
      • Body to be passed
         o File- <<file to be uploaded>>
```

After updating template file, publish can be performed.

Versioning:

The versions of grammar file can be maintained by passing X-Version as mentioned in previous section.

To create a new version of grammar file, first call get grammar

```
GET - {{APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
       • Headers to be passed
         o X-Source-System- <<SourceName>>
         o X-Destination-System- <<DestinationName>>
```

After getting the grammar file, add the changes using all_of openAPI as explained in previous section, after changes are done upload the grammar at specified version. Name the file as <<source>>_<<destination>>.json

```
PUT - {{APIGW}}/mappingSchema/cx/industry/apiIntegration/v1/mappingSchema
     • Headers to be passed:
        o X-Source-System <<sourceName>>
        o X-Target-System <<targetName>>
        o X-Version V1
     • Body to be passed
        o File. <<file to be uploaded>>
```

The destination needs to be updated.

```
PATCH – {{launch_URL}}:443/crmRestApi/resources/11.13.18.05/
atcPublishWorkspaceDestinations
    • Body to be passed
    {
      "DestVersion": "V1"
    }
```

Once grammar is uploaded, you can start publishing.

Supported 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

```
POST - {{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>>
```

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

Update Mapping OAS File

```
PUT - {{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>>
```

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>>
     o X-Version- V1
• 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>>
     o X-Version - V1
• 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 8-5 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.
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.

For additional information on errors on Publish to Siebel CRM, see the *Siebel CRM REST API Guide* https://www.oracle.com/documentation/siebel-crm-libraries.html.

A
Appendix

Setting Default Entities

Table A-1 Setting Default Entities

Entity	Sample Payload	Description

Service Specification	**Siebel-Default-SS**
	
	{
	"id": "Siebel-Default-SS", "name": "Siebel Default SS",
	"description": "Default Service Spec for Sibel Product Spec", "version": "1.0", "lifecycleStatus": "Active", "isBundle":
	false,
	"@type": "ServiceSpecificationOracle", "validFor":
	{ "startDateTime": "2010-06-19T16:42:23.000Z"
	}
	}
		It is mandatory to have a Service Specifocation in Launch to create Service offers.

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description

Product Specification	**Siebel-Default-PS**
	
	{
	"id": "Siebel-Default-PS", "name": "Siebel Default PS",
	"description": "Default Product Spec for Sibel 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", "href": "https://
	{FusionAppsHost}/crmRestApi/atcProductCatalog/
	11.13.18.05/tmf- api/serviceCatalogManagement/v3/
	serviceSpecification/Siebel- ServiceSpec",
	"version": "1.0",
	"@referredType": "ServiceSpecificationOracle",
	"role": "PRIMARY"
	}
	]
	}
		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.

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description

Custom Profile Specification	**Siebel Default Aggregate Discount CPS**
	
	{
	"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",
	"configurable": true, "minCardinality": 0,
	"maxCardinality": 1, "validFor": {
	"startDateTime": "2022-02-22T00:00:00.000Z"
	}
	}
	]
	}
		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.

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description

Custom Profile Specification	**Siebel Discount Matrix CPS**
	
	Siebel Discount Matrix CPS
	
	{
	"id": "Siebel-Default-DiscMat-CPS",
	"name": "Siebel Default Discount Matrice CPS",
	"description": "Default Custom Profile Spec", "href":
	"https://fa-wrpt-pintlabfadev.fa.ocs.oc-
	test.com/crmRestApi/atcProductCatalog/11.13.18.05/
	productCatalogReferenc eManagement/v1/
	customProfileSpecification/Siebel-Default-DiscMat-
	CPS",
	"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",
	"configurable": true, "minCardinality": 0,
	"maxCardinality": 1, "validFor": {
	"startDateTime": "2022-02-22T00:00:00.000Z"
	},
	"customProfileSpecCharValue": [
	{
	"value": "Clinic", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Commercial", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Company", "valueType": "STRING",
	"isDefault": false
	},
	{
	
	"value": "Competing Dealer", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Competing OEM", "valueType": "STRING",
		The migration process is expected to have a custom profile specification with values present in Siebel.

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description

	<pre>"isDefault": false
	},
	{
	"value": "Competitor", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Consultant", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Contact Us", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Contact Manufacturer", "valueType":
	"STRING", "isDefault": false
	},
	{
	"value": "Convention Center", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Corporate Training Center", "valueType":
	"STRING", "isDefault": false
	},
	{
	"value": "Corporate/Transient", "valueType":
	"STRING", "isDefault": false
	},
	{
	"value": "Corporation", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Dealer", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Department", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Department Group", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Military", "valueType": "STRING",
	"isDefault": false</pre>	

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description

	<pre>},
	{
	"value": "QSR", "valueType": "STRING", "isDefault":
	false
	},
	{
	"value": "Ship To", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Convenience Store", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Manufacturer Rep", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "ODM", "valueType": "STRING", "isDefault":
	false
	},
	{
	"value": "Design House","valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "3rd Part Training Center", "valueType":
	"STRING", "isDefault": false
	},
	{
	"value": "All Suite", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Auto/Home Supply Store", "valueType":
	"STRING", "isDefault": false
	},
	{
	"value": "Banking", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Body Shop", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Branch", "valueType": "STRING",
	"isDefault": false
	},</pre>	

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description

	<pre>{
	"value": "Broker", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Business", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Business Customer", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Advertiser", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Central Bank", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Chain Drug", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Chain Food", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Department Store", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Residential", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Customer", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Agency", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Vendor", "valueType": "STRING",
	"isDefault": false
	},
	{</pre>	

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description

	"value": "Committee", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Contract Research
	Organization","valueType": "STRING", "isDefault":
	false
	},
	{
	"value": "Surgery", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Chemist", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Clinical Directorate", "valueType":
	"STRING", "isDefault": false
	},
	{
	"value": "District Health Authority", "valueType":
	"STRING", "isDefault": false
	},
	{
	"value": "Drug Committee", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Hospital Unit", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "Practice", "valueType": "STRING",
	"isDefault": false
	},
	{
	"value": "PBM", "valueType": "STRING", "isDefault":
	false
	},
	{
	"value": "Pharmaceutical Company", "valueType":
	"STRING", "isDefault": false
	}
	]
	},
	{
	"name": "Product Offering", "valueType":
	"PRODUCT_OFFER", "@type": "ProductOfferingOracle",	

Table A-1 (Cont.) Setting Default Entities

Entity	Sample Payload	Description
	"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, run PATCH method call on 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.