THE POWER TO KNOW.

SAS® Offer Optimization for Communications 5.2 (First Maintenance Release)

Administrator's Guide

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc 2011. *SAS® Offer Optimization for Communications 5.2 (First Maintenance Release): Administrator's Guide* . Cary, NC : SAS Institute Inc.

SAS® Offer Optimization for Communications 5.2 (First Maintenance Release): Administrator's Guide

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, February 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

Contents

About This Book

Audience

This document is primarily intended for administrators who will perform the administrative tasks on a regular basis after the initial installation and configuration of SAS Offer Optimization for Communications. The document gives an overview of the SAS Offer Optimization for Communications architecture. It also explains various architecture components and the interactions between them. In addition, this document provides the data flow diagram. This diagram explains the data processing infrastructure that supports various reporting and analytical exploitations and user-interface-driven analysis. The component-specific administrative tasks are detailed in the respective chapters.

Prerequisites

Before you administer SAS Offer Optimization for Communications, make sure that you are familiar with the following concepts:

basic concepts and components of the SAS Intelligence Platform
: For details, see *SAS Intelligence Platform: Overview*, which is located at **http://support.sas.com/documentation/cdl/en/biov/63145/PDF/default/biov.pdf**.

the SAS environment
: For details, see *SAS Intelligence Platform: System Administration Guide*, which is located at **http://support.sas.com/documentation/cdl/en/bisag/60945/PDF/default/bisag.pdf**.

the SAS applications servers that are required for particular content
: For details, see *SAS Intelligence Platform: Application Server Administrative Guide*, which is located at **http://support.sas.com/documentation/cdl/en/biasag/61237/PDF/default/biasag.pdf**.

security concepts
: You should be familiar with the authentication and authorization concepts. You should also know how to manage access in the metadata layer. In addition, you should know how to create and manage user and group definitions in metadata. For details, see *SAS Intelligence Platform: Security Administration Guide*, which is located at **http://support.sas.com/documentation/cdl/en/bisecag/61133/PDF/default/bisecag.pdf**.

the middle-tier environment
: You should know how to configure the Application server.

SAS products
: You should know the basic procedures for using the applications that you plan to administer. For example, if you are responsible for administering SAS Web Report Studio, then you should know how to log on, navigate, and create reports in SAS Web Report Studio.

server context
: You should have complete information about the SAS Application Server context. A SAS Application Server knows its server context (the context in which it is being used) and makes decisions based on that knowledge. For example, a client such as SAS Data Integration Studio is assigned a default SAS Application Server. When the client generates code, it submits the code to that application server. The application server determines what type of code is being submitted and directs it to the correct server. That is, if the code is a typical SAS code that can be run in SAS Display Manager, the code is executed by the application server's workspace server. In addition, data-related objects such as SAS libraries, database libraries, and OLAP schemas can be assigned to a SAS application server. After this assignment, a client might need to access data in a particular library or OLAP schema. The client then uses a server component that belongs to the application server to which the library or schema is assigned.

Part 1

Introduction

Chapter 1

Architecture Overview

SAS Offer Optimization for Communications Architecture

Overview

The SAS Offer Optimization for Communications architecture is designed to efficiently process a large volume of data. At the same time, the architecture enables the solution to use this data to support user-driven workflow through the application user interface. SAS Offer Optimization for Communications has an n-tier architecture that separates the workflow-related activities from data-intensive process routines and distributes functionality across computer resources that are best suited for these tasks.

You can scale the architecture to meet the demands of your workload. For a large organization, the tiers can be installed across a multitude of machines with different operating systems. For tasks such as developing prototypes and presenting demonstrations, all the tiers can be installed on a single machine. Similarly, if you are implementing SAS Offer Optimization for Communications for small enterprises, then you can install all the tiers on a single machine.

The SAS Offer Optimization for Communications architecture consists of the following four tiers:

Data Tier
: The data tier stores your enterprise data. This data is stored in Oracle tables in appropriate schemas and is used in various analysis-through-client programs.

Server Tier
: The server tier of SAS Offer Optimization for Communications consists of data routines and SAS servers that process your enterprise data based on requests from client programs (through middle-tier services) and other programs.

Middle Tier

The middle-tier of SAS Offer Optimization for Communications provides an environment in which the SAS Offer Optimization for Communications client, along with other business intelligence Web applications, such as SAS Web Report Studio and SAS Information Delivery Portal, can execute in an integrated environment. These applications run in a Web application server and communicate with the user by sending and receiving data from the user's Web browser. The middle-tier applications depend on the servers that are deployed on the server tier to process, query on, and analyze data.

Client Tier

Clients in SAS Offer Optimization for Communications include Web-based and desktop user interface content and applications. These clients provide access to content, appropriate query and reporting interfaces, and business intelligence functionality, including advanced design and analysis tasks for all information consumers in your enterprise.

Application Architecture

The following diagram describes the components in the SAS Offer Optimization for Communications architecture.

Figure 1.1 *SAS Offer Optimization for Communications Architecture*

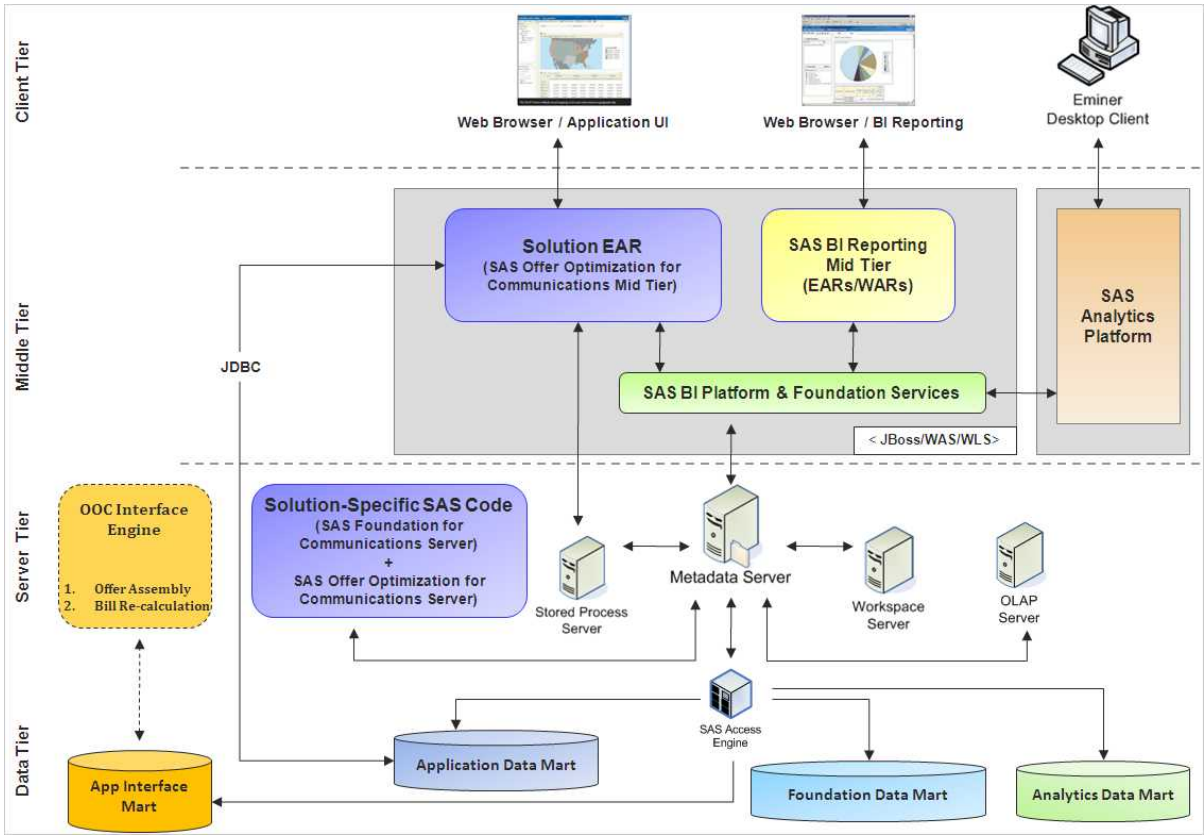


Table 1.1 *Data Tier Components*

Component	Function
Foundation Data Mart	stores your enterprise data (and history) that is classified into reference data, dimensions, and transaction summary facts. This data is used by the business solutions to support analysis at customer, product, and offer level. This data is also used to generate business intelligence reports and analyze the best offer recommendations.
Analytics Data Mart	consists of base data structures that are used to build analytical models. These models are used for scoring or segmenting customers. The solution has a defined set of derived, behavior, and descriptive variables. These variables are initially used to configure the model and later as an input to the scoring code that is provided by the model.
Application Data Mart	stores data for projects that are defined through the application interface to derive best offer recommendations for user-defined customer target population.
Application Interface Mart	contains application data structures that are exposed as a part of the interface to the third-party engines to support the offer evaluation and bill recalculation functionality.

For information about the entities and attributes of each data mart, see *SAS Offer Optimization for Communications: Data Dictionary*.

Table 1.2 *Server Tier Components*

Component	Function
Solution-specific SAS code	represents the data processing SAS routines that are packaged as a part of the solution. These routines perform distinct data operations based on client requests and other SAS routines (in batch mode).
SAS Metadata Server	is a multi-user-centralized resource for storing, managing, and delivering metadata for all SAS applications across your enterprise.
SAS Workspace Server	provides access to SAS software features such as SAS language, SAS libraries, the server file system, results content, and formatting services–execution environments for solution data routines.

Component	Function
SAS Stored Process Server	responds to client requests to execute solution-specific stored processes.
SAS OLAP Server	provides access to solution-specific cubes.

Table 1.3 *Middle-Tier Components*

Component	Function
SAS BI Platform and Foundation Services	consists of SAS Shared Services, SAS Remote Services, Java Platform Services, and SAS Web Infrastructure Platform. For details, see *SAS Intelligence Platform: Overview*, which is located at **http://support.sas.com/documentation/cdl/en/biov/63145/PDF/default/biov.pdf**.
SAS BI Reporting Mid-Tier	provides an execution environment for the following business intelligence applications: • SAS Web Report Studio • SAS Information Delivery Portal • SAS BI Dashboard • SAS BI Portlets
SAS Analytics Platform	provides a common application framework for analytical applications such as SAS Enterprise Miner and SAS Forecast Server. For details, see *Administrator's Guide for SAS Analytics Platform*, which is located at **http://support.sas.com/documentation/onlinedoc/apcore/admin15.pdf**.
SAS Offer Optimization for Communications Middle Tier	consists of solution-specific services that interact with the client interface to accept user requests (query analysis or data processing) and respond to them with the help of the server tier.

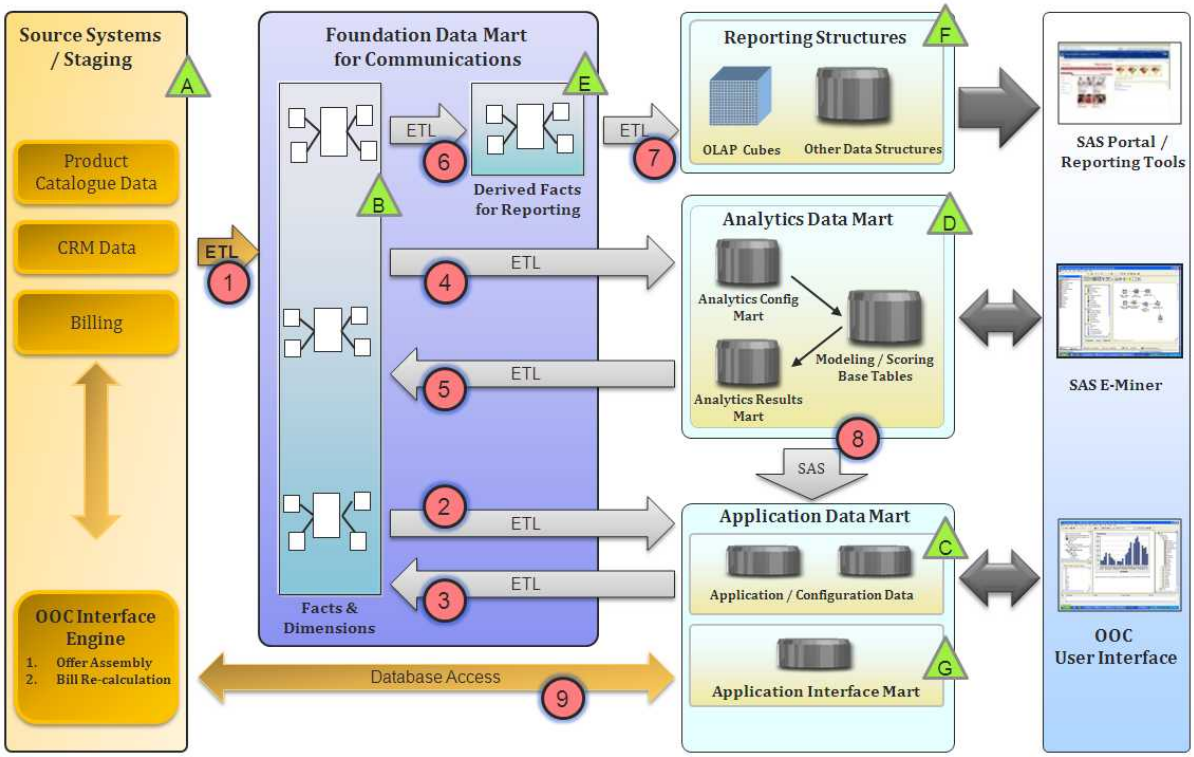
Table 1.4 *Client-Tier Components*

Component	Function
BI Reporting Interface	The BI Reporting Interface mainly consists of the following client interfaces: • SAS OLAP Cube Studio • SAS Information Map Studio • SAS Web Report Studio • SAS Information Delivery Portal • SAS BI Dashboard • SAS BI Portlets For information and usage of these clients, see *SAS Intelligence Platform: Overview*, which is located at `http://support.sas.com/documentation/cdl/en/biov/63145/PDF/default/biov.pdf`.
SAS Enterprise Miner desktop client and SAS Enterprise Guide	For information and usage of these clients, see *SAS Intelligence Platform: Overview*, which is located at `http://support.sas.com/documentation/cdl/en/biov/63145/PDF/default/biov.pdf`.
SAS Offer Optimization for Communications Web User Interface	is the Web-based user interface to support configuration and execution of user-driven workflow for recommending best offers. For details, see *SAS Offer Optimization for Communications: User's Guide*, which is located at `http://support.sas.com/documentation/solutions/offeropt/index.html`.

Data Flows in SAS Offer Optimization for Communications

The following diagram represents the data stores that form the solution information model and the data flow through this information model. As depicted in the diagram data flows through the following data stores:

- A: Source Systems
- B: Foundation Data Mart
- C: Application Data Mart
- D: Analytics Data Mart
- E: Derived Facts for Reporting
- F: Reporting Structures



The data flows in SAS Offer Optimization for Communications are handled through the following ETLs:

1 ETLs that populate the Foundation data mart from the source systems structures. These ETLs are typically developed during implementation with additional information such as inputs on data availability, reference patterns, information validations, refresh frequency, and so on.

2 ETLs that populate customer, offer, usage, and revenue information into the Application data mart.

3 ETLs that write back the business group and customer association data into the Foundation data mart. This association information is further used in all aspects of the solution such as UI-based analysis, analytics, and business reporting.

4 ETLs that populate the base tables for supporting solution analytics. The base tables are used to build analytical models, which are used for scoring or segmenting customers or subjects of analysis.

5 ETLs that write back the results of analytical or scoring processes into the Foundation data mart.

6 ETLs that extract and aggregate transaction data according to the reporting grain of the solution and populate into reporting facts.

7 ETL jobs that populate data from the derived facts and dimensions (Foundation data mart) into the solution cubes.

8 SAS data routines that extract analytical information. This information is used in the application workflow.

9 Denotes the database access process, which is used by the SAS Offer Optimization for Communications interface engine to process data from and into the Application Interface data mart

SAS Offer Optimization for Communications contains three data flows:

BI Reporting data flow
For details, see "ETL Jobs for BI Reports" on page 25.

Analytics data flow
For details, see "ETL Jobs for Analytics" on page 35.

Application data flow
For details, see "ETL Jobs for Populating Application Data Mart" on page 151.

The SAS Offer Optimization for Communications Workflow

The SAS Offer Optimization for Communications workflow explains the interactions among various components. To summarize, the solution workflow contains the following steps:

1. Populate data into the common data layer from the external source systems through the staging area.
2. Populate data into the solution-specific data layer:
   - application-specific data
   - analytical data
   - reporting data

   Note: For details about steps 1 and 2, see the relevant chapters of this guide.
3. Log on to SAS Offer Optimization for Communications with the profile of an administrator.
   a. Define business groups.
   b. Run process to add customers to business groups.

      Note: For tasks that are detailed in step 3, see *SAS Offer Optimization for Communications: User's Guide.*
4. Log on to SAS Enterprise Miner with a certain profile.
   a. Build and register an analytical model to analyze customer churn.
   b. Build and register an analytical model to analyze customer segmentation.
   c. Build and register an analytical model to analyze cross-sell and up-sell.
5. Run back-end processes.
   a. Create segments of the business groups.
   b. Generate churn scores for each customer.
   c. Generate cross-sell and up-sell scores.

      Note: For tasks that are detailed in step 4 and 5, see the relevant chapters of this guide.
6. View business reports in SAS Web Report Studio.
   a. Analyze churn reports.
   b. Analyze segmentation reports.
   c. Analyze cross-sell and up-sell reports.

d. Analyze business groups reports.

e. Identify business problems associated with each business group.

7. Log on to SAS Offer Optimization for Communications with a certain profile.

   a. Define projects with specific objectives for different business groups.

   b. Configure and run project workflow to derive representative customers.

   c. Export information about representative customers to external source systems.

   d. Import billing details of representative customers and recalculate invoices.

   e. Produce best offers for customers in the target segment.

   f. Promote the project to batch mode.

   Note: For tasks that are detailed in steps 6 and 7, see *SAS Offer Optimization for Communications: User's Guide*.

8. Run the project in batch mode and produce best offers for each customer in the customer base.

9. Export information about best offers to external source systems.

   Note: For tasks that are detailed in step 8 and 9, see the relevant chapter of this guide.

10. Log on to SAS Web Report Studio with a certain profile and view reports to evaluate the performance of SAS Offer Optimization for Communications.

    Note: For details, see *SAS Offer Optimization for Communications: User's Guide*.

Part 2

Application Management

Chapter 2

Configuring the Environment for ETL Jobs

Grant Random Access for Data in Defined Libraries

Some ETL jobs might require random access to the data and therefore might not run successfully. Therefore, in order to ensure that these ETLs run successfully, you have to grant random access for data in the libraries.

To grant random access to data:

1. Log on to SAS Data Integration Studio and connect to a profile.
2. Select the **Inventory** tab and expand the **Library** folder.
3. Right-click the **BPPAPTBL** library and select **Properties**.
   a. On the **Options** tab, click **Advanced Options**.
   b. On the **Input/Output** tab, select `Yes` for **Allow random access to a table when rereading a row**.
   c. Click **OK**.
4. Click **OK**.
5. Repeat steps 3 and 4 for all the other library definitions.
6. Close SAS Data Integration Studio.

Initialize Parameters for ETL Jobs

A sample script is packaged in order to initialize parameters that are required for various ETL jobs and data structures such as OLAP cubes and information maps. As a result, the CFDMISC.SETUP_PARAM table is populated with default values. For details, see *SAS Offer Optimization of Communications: Installation and Configuration Guide*. This document is available at the following location: **http://support.sas.com/documentation/solutions/offeropt/index.html**.

You can change these values according to your installation requirements.

To change the parameter values:

1. In the CFDMISC.SETUP_PARAM table, set up the parameter values according to your requirements. For details, see "Configuring the Setup_Param Table" on page 15.
2. Log on to SAS Data Integration Studio and connect to a profile.
3. Expand **Products** ⇨ **SAS Foundation for Communications** ⇨ **Jobs** ⇨ **Setup Jobs**.
4. Run the cfd_param_declaration_job job. This step generates the setup_param.sas file in the **<SAS configuration path>/Lev1/Applications/SASFoundForComm5.2/Data/cfdparam** folder.
5. Close SAS Data Integration Studio.

Create Catalogs for User-Defined Formats

To create catalogs for user-defined formats:

1. Go to the **<SAS configuration path>/Lev1/Applications/SASFoundForComm5.2/Data/format** folder.
2. Make sure that the following data sets are available:
   - FMTLIST
   - FMTBANDLIST
3. In both these data sets, set the value of the LIBN column to the owner name of the schema that is created for reference tables. For example, you can specify the value as *REF*.
4. Open Base SAS.
5. Declare the LIBNAME statement for the schema that is created for reference tables.
   a. Make sure that you make the declaration in the same session in which the fmt_gen macro is run.
   b. Set the value of the LIBN column and the LIBNAME as explained in the example below:

```
LIBNAME REF oracle user=REF password=REF PATH=TESTDB;
```

6. Run the comfdnsrvc_autoexec.sas code that is located in the **<SAS configuration path>/Lev1/SASApp**. This code initializes the required parameters.
7. Run the fmt_gen and prepare_bands_formats macros. These macros are located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASFoundForComm5.2/SASMacro** folder.

   Note: The formats are based on the reference data. Therefore, if the data in the reference tables is updated, you have to refresh the formats also. To do so, repeat this step.
8. Close Base SAS.

Configuring the Setup_Param Table

The Setup_Param table contains parameters that you have to configure for ETL jobs. These parameters are initialized when you run the INSERT script to populate the Setup_Param table. You can update this table when you want to change the value of any of the parameters.

The following table lists parameters that you can configure in the Setup_Param table.

Table 2.1 *Setup_Param Table*

Parameter Name	Parameter Description	Purpose
run_mth_strt_dt	Run Month Start Date	contains the starting date for the run month. The date is stored in the DDMONYYYY format. For regularly scheduled loads, this date is the starting date of a calendar month. For example, if a job is scheduled in January 2010, then the value of this parameter is 01JAN2010. For scheduled loads, the fd_param_monthly_dt_update_job job updates the value for this parameter.
run_mth_end_dt	Run Month End Date	contains the ending date for the run month. The date is stored in the DDMONYYYY format. For regularly scheduled loads, this date is the ending date of a calendar month. For example, if a job is scheduled in January 2010, then the value of this parameter is 31JAN2010. For scheduled loads, the cfd_param_monthly_dt_update_job job updates the value for this parameter.
global_high_dt	Global High Date	contains the global high date in the DDMONYYYY format. This date can be a future date such as 31JAN9999.

Parameter Name	Parameter Description	Purpose
run_mth_strt_dt_sk	Run Month Start Date Surrogate Key	contains the surrogate key of the starting date of the run month. For regularly scheduled loads, this date stores the surrogate key for the starting date of a calendar month. For example, if a job is scheduled in January 2010, then the value of the parameter is the surrogate key of 01JAN2010. For scheduled loads, the cfd_param_monthly_sk_update_job job updates the value for this parameter.
run_mth_end_dt_sk	Run Month End Date Surrogate Key	contains the surrogate key of the ending date of the run month. For regularly scheduled loads, this date stores the surrogate key for the calendar month end date. For example, for a job scheduled on Jan 2010, the value of the parameter would be the surrogate key corresponding to 31JAN2010. For scheduled loads, the cfd_param_monthly_sk_update_job job updates the value for this parameter.
run_wk_strt_dt	Run Week Start Date	contains the starting date of the run week in the DDMONYYYY date format. For regularly scheduled loads, this date is the first day of the calendar week. For scheduled loads, the cfd_param_weekly_dt_update_job job updates the value for this parameter.
run_wk_end_dt	Run Week End Date	contains the ending date of the run week in the DDMONYYYY format. For regularly scheduled loads, this date is the last day of a calendar week. For scheduled loads, the cfd_param_weekly_dt_update_job job updates the value for this parameter.
run_wk_strt_dt_sk	Run Week Start Date Surrogate Key	contains the surrogate key of the starting date of the run week. For regularly scheduled loads, this date is the surrogate key of the first day of the calendar week. For scheduled loads, the cfd_param_weekly_sk_update_job job updates the value for this parameter.

Parameter Name	Parameter Description	Purpose
run_wk_end_dt_sk	Run Week End Date Surrogate Key	contains the surrogate key of the ending date of the run week. For regularly scheduled loads, this date is the surrogate key of the last day of the calendar week. For scheduled loads, the cfd_param_weekly_sk_update_job job updates the value for this parameter.
bill_mth_strt_dt	Bill Month Start Date	contains the starting date of the billing month in the DDMONYYYY date. For scheduled loads, the cfd_param_bill_cycle_dt_job job updates the value for this parameter.
bill_mth_end_dt	Bill Month End Date	contains the ending date of the billing month in the DDMONYYYY format. For scheduled loads, the cfd_param_bill_cycle_dt_job job updates the value for this parameter.
bill_mth_strt_dt_sk	Bill Month Start Date Surrogate Key	contains the surrogate key of the starting date of the billing month. For scheduled loads, the cfd_param_bill_cycle_sk_job job updates the value for this parameter.
bill_mth_end_dt_sk	Bill Month End Date Surrogate Key	contains the surrogate key of the ending date of the billing month. For scheduled loads, the cfd_param_bill_cycle_sk_job job updates the value for this parameter.
inquirytyp_billing	Type Code for Inquiry on Billing	specifies the value for the inquiry type code for billing-related inquires.
complnttyp_billing	Type Code for Complaint on Billing	specifies the value for the complaint type code for billing-related complaints.
run_bill_cyc_cd	Run Bill Cycle Code	specifies the value for the billing cycle code for the current run. For scheduled loads, the cfd_param_bill_cycle_cd_job job updates the value for this parameter.
yes_flag	Yes Flag	specifies the character value for the flag that indicates Yes. This parameter is configured only once.

Parameter Name	Parameter Description	Purpose
navl_sk	Not Available Surrogate Key	specifies the surrogate key of the default records in the dimension tables. The same value is added to the fact tables for records where a value of a particular dimension value is missing. This parameter is configured only once and generally has a negative value, which is not figured in the surrogate key sequence. For example, the value for this parameter can be -999.
pre_pd_offer_pymnt_mode_cd	Prepaid Offer Payment Mode Code	specifies the offer payment mode code that identifies prepaid offers.
no_flag	No Flag	specifies the character value for the flag that indicates No. This parameter is configured once.
navl_measure	Not Available Measure	specifies the default value for measures. This parameter is configured only once and generally has a value of zero (numeric).
no_num_flag	No Num Flag	specifies the numeric value for the flag that indicates No. This parameter is configured only once.
yes_num_flag	Yes Num Flag	specifies the numeric value for the flag that indicates Yes. This parameter is configured only once.
no_of_day_in_mth	Number of Days in Month	specifies the number of days in a month.
language_cd	Language Code	specifies the default locale code. Reference tables enable you to store descriptions in multiple languages. Therefore, if the implementation is in US English, then it will contain en_US.
data	Data	specifies the service category code for DATA type of services.
voice	Voice	specifies the service category code for VOICE type of services.
message	Message	specifies the service category code for MESSAGE type of services.
navl_cd	Not Available Code	specifies the code for the default value of the character columns.

Parameter Name	Parameter Description	Purpose
srvc_active	Service Active Code	specifies the code for status of the services that are active.
srvc_closed	Service Closed Code	specifies the code for status of the services that are closed.
lst_run_mth_end_dt_sk	Last Run Month End Date Surrogate Key	contains the surrogate key of the ending date of the month prior to the current run month. For scheduled loads, the cfd_param_monthly_sk_update_job job updates the value for this parameter.
lst_run_wk_end_dt_sk	Last Run Week End Date Surrogate Key	contains the surrogate key of the ending date of the week prior to the current run week. For scheduled loads, the cfd_param_weekly_sk_update_job job updates the value for this parameter.
lst_mth_run_bill_cyc_end_dt_sk	Last Month Run Bill Cycle End Date Surrogate Key	contains the surrogate key of the ending date of the billing month, which is prior to the current billing month. For scheduled loads, the cfd_param_bill_cycle_sk_job job updates the value for this parameter.
lst2_run_mth_end_dt_sk	Last Two Run Month End Date Surrogate Key	contains the surrogate key of the ending date of two months prior to the current run month. For scheduled loads, the cfd_param_monthly_sk_update_job job updates the value for this parameter.
lst2_run_mth_strt_dt	Last Two Run Month Start Date	contains the starting date of two months prior to the current run month. For scheduled loads, the cfd_param_monthly_dt_update_job job updates the value for this parameter.
lst2_run_wk_end_dt_sk	Last Two Run Week End Date Surrogate Key	contains the surrogate key of the ending date of two weeks prior to the current run week. For scheduled loads, the cfd_param_weekly_sk_update_job job updates the value for this parameter.
bg_mvmnt_cat_grp_cd	BG Movement Category Group Code	specifies the category group code for customer movements across business groups.
sgmt_mvmnt_cat_grp_cd	Segment Movement Category Group Code	specifies the category group code for customer movements across segments.

Parameter Name	Parameter Description	Purpose
lst_run_mth_end_dt	Last Run Month End Date	contains the ending date of the month that is prior to the current run month. For scheduled loads, the cfd_param_monthly_dt_update_job job updates the value for this parameter.
cfdn_model_churn	Common Foundation Model Type Code for Churn	specifies the code for an analytical model that is of the churn type.
cfdn_model_sgmt	Common Foundation Model Type Code for Segmentation	specifies the code for an analytical model that is of the segmentation type.
lst2_run_mth_end_dt	Last Two Run Month End Date	contains the ending date of the month, which is two months prior to the current run month. For scheduled loads, the cfd_param_monthly_dt_update_job job updates the value for this parameter.
lst2_run_mth_strt_dt_sk	Last Two Run Month Start Date Surrogate Key	contains the surrogate key of the starting date of the month, which is two months prior to the current run month. For scheduled loads, the cfd_param_monthly_sk_update_job job updates the value for this parameter.
lst_run_wk_end_dt	Last Run Week End Date	contains the ending date for the week prior to the current run week. For scheduled loads, the cfd_param_weekly_dt_update_job job updates the value for this parameter.
lst2_run_wk_end_dt	Last Two Run Week End Date	contains the ending date for the week, which is two weeks prior to the current run week. For scheduled loads, the cfd_param_weekly_dt_update_job job updates the value for this parameter.
lst2_run_wk_strt_dt	Last Two Run Week Start Date	contains the starting date for the week, which is two weeks prior to the current run week. For scheduled loads, the cfd_param_weekly_dt_update_job job updates the value for the parameter.
lst2_run_wk_strt_dt_sk	Last Two Run Week Start Date Surrogate Key	contains the surrogate key of the ending date of the week, which is two weeks prior to the current run week. For scheduled loads, the cfd_param_weekly_sk_update_job job updates the value for this parameter.

Parameter Name	Parameter Description	Purpose
cfdn_dim_owner_nm	Common Foundation Dim Owner Name	specifies the name of the user who owns the schema that is created for dimension tables.
cfdn_misc_owner_nm	Common Foundation Misc Owner Name	specifies the name of the user who owns the schema that is created for miscellaneous tables.
cfdn_fact_owner_nm	Common Foundation Fact Owner Name	specifies the name of the user who owns the schema that is created for fact tables.
cfdn_ref_owner_nm	Common Foundation Ref Owner Name	specifies the name of the user who owns the schema that is created for reference tables.
cfdn_base_owner_nm	Common Foundation Base Owner Name	specifies the name of the user who owns the schema that is created for base tables.
pymt_stts_bncd	Bounced Payment Status	stores the bounced payment status
chng_rsn_vlnt	Voluntary	indicates whether activation or deactivation was done voluntarily or involuntarily.
cfdn_model_churn		identifies whether the type of model is customer retention.

Setup Jobs

The ETL processes that populate the Foundation data mart require initialization and metadata registration of solution-specific parameters. The setup jobs enable you to configure the environment according to these requirements. You have to run these jobs before every schedule. The following table lists the setup jobs that you have to run in SAS Data Integration Studio.

Table 2.2 *Setup Jobs*

Job Name	Purpose	Primary Source Table	Target Table
cfd_param_declaration_job	declares global variables based on their data types (for example, Character, Number, and Date) and also initializes their values.	SETUP_PARAM	setup_param.sas (File)

Job Name	Purpose	Primary Source Table	Target Table
cfd_param_bill_cycle_dt_job	increments dates related to a billing cycle to their subsequent values based on the current billing cycle and the subsequent billing cycle.	SETUP_PARAM	SETUP_PARAM
cfd_param_bill_cycle_sk_job	increments surrogate keys for dates related to a billing cycle to their subsequent values based on the current billing cycle and the subsequent billing cycle.	SETUP_PARAM	SETUP_PARAM
cfd_param_bill_cycle_cd_job	updates the current billing cycle code to the subsequent billing cycle code.	SETUP_PARAM	SETUP_PARAM
cfd_param_monthly_dt_update_job	increments the run dates of calendar month to their subsequent values.	SETUP_PARAM	SETUP_PARAM
cfd_param_monthly_sk_update_job	increments the surrogate keys for run dates of a calendar month to their subsequent values.	SETUP_PARAM	SETUP_PARAM
cfd_param_weekly_dt_update_job	increments the run dates of a calendar week to their subsequent values.	SETUP_PARAM	SETUP_PARAM
cfd_param_weekly_sk_update_job	increments the surrogate keys for run dates of a calendar week to their subsequent values.	SETUP_PARAM	SETUP_PARAM

ETL Jobs for Eligibility Criteria

The ETL jobs for eligibility criteria are provided as optional jobs. You can use these jobs if you do not have your own definition of the eligibility criteria and a specific process to track them. In addition, you can use these jobs if you want to use eligibility as a dimension for analysis purpose. However, if you have well-defined eligibility criteria, then you must populate the CUST_X_ELIGIBILITY_BRIDGE and ELIGIBILITY_COMBINATION_D tables from the source system during implementation by developing the appropriate ETL jobs..

Table 2.3 *ETL Jobs for Eligibility Criteria*

Job Name	Purpose	Primary Source Table	Target Table
cfd_eligibility_config_job	uses distinct values of the attributes of the ELIGIBILITY_ATTRBT_D table and creates the eligibility combinations.	• CUST_D • ELIGIBILITY_ATTRBT_D	• ELIGIBILITY_COMBINATION_DTL • ELIGIBILITY_COMBINATION_D
cfd_cust_eligibility_tagging_job	associates the customer to an eligibility combination, based on the eligibility attributes.	• CUST_D • ELIGIBILITY_ATTRBT_D	• CUST_X_ELIGIBILITY_BRIDGE • ELIGIBILITY_COMBINATION_DTL • ELIGIBILITY_COMBINATION_D

Chapter 3

ETL Jobs for BI Reports and Analytics

Processing Jobs

The processing jobs are mandatory jobs. You have to run these jobs before you run any BI and analytics jobs.

Table 3.1 *Processing Jobs*

Job Name	Purpose	Primary Source Table	Target Table
cfd_pre_pd_cust_tmp_job	extracts values for the customer_id and customer_sk columns for all prepaid customers who are active during a certain week.	CUST_D	PRE_PD_CUST_TMP
cfd_pst_pd_cus_x_bill_cycle_bridge_tmp_job	extracts values for the customer_id and customer_sk columns for all postpaid customers who are active during a certain billing cycle window.	CUST_D	CUST_X_BILL_CYCLE_BRIDGE_TMP

ETL Jobs for BI Reports

The ETL jobs for BI (Business Intelligence) reports populate the fact tables that are required for generating BI reports. For details about BI reports, see the *SAS Offer Optimization for Communications: User's Guide*.

Based on the aggregation of data, these jobs are categorized as monthly, weekly, and bill-monthly. The following table lists the ETL jobs for BI reports that you have to run in SAS Data Integration Studio.

Table 3.2 *ETL Jobs for BI Reports*

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_bill_monthly_d_tmp_job	populates customer-related dimensional data for customers who are active during a certain billing cycle. This data mostly includes details such as codes and indicators. The data in the CUST_BILL_MTH_D_TMP target table is subsequently reused in the final data load of the cust_mth_summary_f and cust_service_mth_f tables.	CUST_D	CUST_BILL_MTH_D_TMP
cfd_cust_monthly_d_tmp_job	populates customer-related dimensional data for customers who are active during a certain calendar month cycle. This data mostly includes details such as codes and indicators. The data in the CUST_BILL_MTH_D_TMP target table is subsequently reused in the final data load of the cust_bill_mth_summary_f and cust_service_mth_f tables.	CUST_D	CUST_MTH_D_TMP
cfd_cust_weekly_d_tmp_job	populates customer-related dimensional data for customers who are active during a certain calendar week cycle. This data mostly includes details such as codes and indicators. The data in the CUST_BILL_MTH_D_TMP target table is subsequently reused in the final data load of the cust_mth_summary_f and cust_service_mth_f tables.	CUST_D	CUST_WEEKLY_D_TMP

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_bill_monthly_f_tmp_job	populates fact-related data for customers who are active during a certain billing cycle . The data in the CUST_BILL_MTH_F_TMP target table is subsequently reused in the final data load of the cust_bill_mth_summary_f table.	USAGE_SUMMARY_F	CUST_BILL_MTH_F_TMP
cfd_cust_monthly_f_tmp_job	populates fact-related data for customers who are active during a certain calendar month. The data in the CUST_MTH_F_TMP target table is subsequently reused in the final data load of the cust_mth_summary_f table.	USAGE_SUMMARY_F	CUST_MTH_F_TMP
cfd_cust_weekly_f_tmp_job	populates fact-related data for customers who are active during a certain calendar week. The data in the CUST_WEEKLY_F_TMP target table is subsequently reused in the final data load of the cust_weekly_summary_f table.	USAGE_SUMMARY_F	CUST_WEEKLY_F_TMP
cfd_cust_service_weekly_f_tmp_job	populates fact-related data at a product and service level for the customers who are active during a certain calendar week. The data in the CUST_SERVICE_WEEKLY_F_TMP target table is subsequently reused in the final data load of the cust_service_weekly_f table.	• USAGE_SUMMARY_F • SUBSCRP_SERVICE_ACTIVITY_F	CUST_SERVICE_WEEKLY_F_TMP

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_service_bill_mth_f_tmp_job	populates fact-related data at a product and service level for the customers who are active during a certain billing cycle. The data in the CUST_SERVICE_BILL_MTH_F_TMP target table is subsequently reused in the final loading of the cust_service_bill_mth_f table.	• USAGE_SUMMARY_F • SUBSCRP_SERVICE_ACTIVITY_F	CUST_SERVICE_BILL_MTH_F_TMP
cfd_cust_service_mth_f_tmp_job	populates fact-related data at a product and service level for the customers who are active during a certain calendar month. The data in the CUST_SERVICE_MTH_F_TMP target table is subsequently reused in the final data load of the cust_service_mth_f table.	• USAGE_SUMMARY_F • SUBSCRP_SERVICE_ACTIVITY_F	CUST_SERVICE_MTH_F_TMP
cfd_cust_prev_bill_mth_arpu_tmp_job	calculates values for various ARPU types such as DATA, VOICE, and GROSS. This job also calculates tenure of customers who are active during a certain billing cycle. These values are calculated based on the data that is available until the starting date of a certain billing cycle . This table is subsequently used to calculate ARPU and Tenure until the ending date of a certain billing cycle period through a bootstrapping mechanism.	CUST_BILL_MTH_SUMMARY_F	CUST_PREV_BILL_MTH_ARPU_TMP

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_service_prev_mth_arpu_tmp_job	calculates values for various ARPU types such as DATA, VOICE, and GROSS. This job also calculates the tenure of customers who are active during a certain calendar month at a product and service level. These values are calculated based on the data that is available until the starting date of a certain calendar month. This table is subsequently used to calculate ARPU and tenure until the ending date of a certain calendar month through a bootstrapping mechanism.	CUST_SERVICE_MTH_F	CUST_SERVICE_PREV_MTH_ARPU_TMP
cfd_cust_service_prev_bill_mth_arpu_tmp_job	calculates values for various ARPU types such as DATA, VOICE, and GROSS. This job also calculates the tenure of customers who are active during a certain billing cycle at a product and service level. These values are calculated based on the data that is available until the starting date of a certain billing cycle period. This table is subsequently used to calculate the ARPU and tenure until the ending date of a certain billing cycle through a bootstrapping mechanism.	CUST_SERVICE_BILL_MTH_F	CUST_SERVICE_BILL_MTH_ARPU_TMP

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_service_prev_weekly_arpu_tmp_job	calculates various ARPU types such as DATA, VOICE, and GROSS. This job also calculates tenure of customers who are active during a certain calendar week at a product and service level. These values are calculated based on the data that is available until the starting date of a certain calendar week. This table is subsequently used to calculate the ARPU and tenure until the ending date of a certain calendar week through a bootstrapping mechanism.	CUST_SERVICE_WEEKLY_F	CUST_SERVICE_WEEKLY_ARPU_TMP
cfd_cust_prev_mth_arpu_tmp_job	calculates various ARPU types such as DATA, VOICE, and GROSS. This job also calculates tenure of customers who are active during a certain calendar month. These values are calculated based on the data that is available until the starting date of a certain calendar month. This table is subsequently used to calculate the ARPU and Tenure until the ending date of a certain calendar month through a bootstrapping mechanism.	CUST_MTH_SUMMARY_F	CUST_PREV_MTH_ARPU_TMP
cfd_cust_prev_weekly_arpu_tmp_job	calculates various ARPU types such as DATA, VOICE, and GROSS. This job also calculates tenure of customers who are active during a certain calendar week. These values are calculated based on the data that is available until the starting date of a certain calendar week. This table is subsequently used to calculate the ARPU and tenure until the ending date of a certain calendar week through a bootstrapping mechanism.	CUST_WEEKLY_SUMMARY_F	CUST_PREV_WEEKLY_ARPU_TMP

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_bill_monthly_summary_incr_f_job	populates the Bill Monthly Summary Data relevant for a certain billing cycle only. This data is later used to directly append to the customer bill monthly summary fact table and is also used to update the customer bill monthly cube incrementally.	• CUST_BILL_MTH_D_TMP • CUST_BILL_MTH_F_TMP • CUST_PREV_BILL_MTH_ARPU_TMP	CUST_BILL_MTH_SUMMARY_INCR_TMP
cfd_cust_service_bill_mth_summary_incr_f_job	populates the Customer Service Bill Monthly Summary Data that is relevant for a certain billing cycle. This data is further used to directly append to the customer service bill monthly summary fact table and is also used to update the customer service bill monthly cube incrementally.	• CUST_BILL_MTH_D_TMP • CUST_SERVICE_BILL_MTH_F_TMP • CUST_SERVICE_BILL_MTH_ARPU_TMP	CUST_SERVICE_BILL_MTH_INCR_TMP
cfd_cust_service_monthly_summary_incr_f_job	populates the Customer Service Monthly Summary Data relevant for a certain calendar month. This data is further used to directly append to the customer service monthly summary fact table and is also used to update the customer service monthly cube incrementally.	• CUST_MTH_D_TMP • CUST_SERVICE_MTH_F_TMP • CUST_SERVICE_PREV_MTH_ARPU_TMP	CUST_SERVICE_MTH_INCR_TMP
cfd_cust_service_weekly_summary_incr_f_job	populates the Customer Service Weekly Summary Data relevant for a certain calendar week only. This data is appended to the customer service weekly summary fact table and is also added to the customer service weekly cube incrementally.	• CUST_MTH_D_TMP • CUST_SERVICE_WEEKLY_F_TMP • CUST_SERVICE_WEEKLY_ARPU_TMP	CUST_SERVICE_WEEKLY_INCR_TMP
cfd_cust_monthly_summary_incr_f_job	populates the Monthly Summary Data relevant for a certain calendar month only. This data is appended to the customer monthly summary fact table and is also added to the customer monthly cube incrementally.	• CUST_MTH_D_TMP • CUST_MTH_F_TMP • CUST_PREV_MTH_ARPU_TMP	CUST_MTH_SUMMARY_INCR_TMP

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_weekly_summary_incr_f_job	populates the Weekly Summary Data relevant for a certain calendar week only. This data is appended to the customer weekly summary fact table and also added to the customer weekly cube incrementally.	• CUST_WEEKLY_D_TMP • CUST_WEEKLY_F_TMP • CUST_PREV_WEEKLY_ARPU_TMP	CUST_WEEKLY_SUMMARY_INCR_TMP
cfd_cust_bill_monthly_summary_f_job	appends the customer billing monthly summary data for a certain billing cycle from the CUST_BILL_MTH_SUMMARY_INCR_TMP table to the existing customer billing monthly summary data in the CUST_BILL_MTH_SUMMARY_F fact.	CUST_BILL_MTH_SUMMARY_INCR_TMP	CUST_BILL_MTH_SUMMARY_F
cfd_cust_monthly_summary_f_job	appends the customer monthly summary data for a certain calendar month from the CUST_MTH_SUMMARY_INCR_TMP table to the existing customer monthly summary data in the CUST_MTH_SUMMARY_F fact.	CUST_MTH_SUMMARY_INCR_TMP	CUST_MTH_SUMMARY_F
cfd_cust_service_bill_mth_summary_f_job	appends the customer service bill monthly summary data for a certain billing cycle from the CUST_SERVICE_BILL_MTH_INCR_TMP table to the existing customer service bill monthly summary data in the CUST_SERVICE_BILL_MTH_F fact.	CUST_SERVICE_BILL_MTH_INCR_TMP	CUST_SERVICE_BILL_MTH_F
cfd_cust_service_mth_summary_f_job	appends the customer service monthly summary data for a certain calendar month from the CUST_SERVICE_MTH_INCR_TMP table to the existing customer service monthly summary data in the CUST_SERVICE_MTH_F fact.	CUST_SERVICE_MTH_INCR_TMP	CUST_SERVICE_MTH_F

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_service_weekly_summary_f_job	appends the customer service weekly summary data for a certain calendar week from the CUST_SERVICE_WEEKLY_INCR_TMP table to the existing customer service weekly summary data in the CUST_SERVICE_WEEKLY_F fact.	CUST_SERVICE_WEEKLY_INCR_TMP	CUST_SERVICE_WEEKLY_F
cfd_cust_weekly_summary_f_job	appends the customer monthly summary data for a certain calendar week from the CUST_WEEKLY_SUMMARY_INCR_TMP table to the existing customer weekly summary data in the CUST_WEEKLY_SUMMARY_F fact.	CUST_WEEKLY_SUMMARY_INCR_TMP	CUST_WEEKLY_SUMMARY_F
cfd_cust_cmprtv_monthly_summary_f_job	loads data for customers who opted for the best offer in the last calendar month and are still continuing with the communications service provider with the same offer. For such customers the monthly summary data for the period prior to the period of taking the best offer is juxtaposed with the monthly summary data for the current period.	CUST_MTH_SUMMARY_F	CUST_CMPRTV_MTH_SUMMARY_F
cfd_cust_cmprtv_weekly_summary_f_job	loads data for customers who have opted for best price plan offer in the last calendar week and are still continuing with the communications service provider with the same offer. For such customers the weekly summary data for the period prior to the period of taking the BPP offer is juxtaposed with the weekly summary data for the current period.	CUST_WEEKLY_SUMMARY_F	CUST_CMPRTV_WEEKLY_SUMMARY_F

Job Name	Purpose	Primary Source Table	Target Table
cfd_cust_mth_bg_mvmnt_analysis_f_job	loads data related to the customer movement across business groups. In other words, these are customer counts from a business group in the previous month to the remaining business groups in the current month.	CUST_MTH_SUMMARY_F	CUST_MTH_MOVEMENT_ANALYSIS_F
cfd_cust_mth_sgmnt_mvmnt_analysis_f_job	loads data related to customer segment movement (that is, customer counts from one segment in the previous month to the remaining segments in the current month) for a particular model.	CUST_MTH_SUMMARY_F	CUST_MTH_MOVEMENT_ANALYSIS_F
cfd_cust_monthly_csus_job	loads the latest cross-sell and up-sell scores from the scoring table to the cross-sell and up-sell reporting table. The grain of the data that is populated is monthly and is at customer level.	• PREDICTIVE_MODEL_RUN_DTL • CUST_D	CUST_MTH_CSUS_F
cfd_cust_weekly_csus_job	loads the latest cross-sell and up-sell scores from the scoring table into the cross-sell and up-sell reporting table. The grain of the data that is populated is weekly and is at customer level.	• PREDICTIVE_MODEL_RUN_DTL • CUST_D	CUST_WEEKLY_CSUS_F
cfd_subscrp_monthly_csus_job	loads the latest cross-sell and up-sell scores from the scoring table into the cross-sell and up-sell reporting table. The grain of the data that is populated is monthly and is at subscription level.	• PREDICTIVE_MODEL_RUN_DTL • SUBSCRP_D	SUBSCRP_MTH_CSUS_F
cfd_subscrp_weekly_csus_job	loads the latest cross-sell and up-sell scores from the scoring table into the cross-sell and up-sell reporting table. The grain of the data that is populated is weekly and is at subscription level.	• PREDICTIVE_MODEL_RUN_DTL • SUBSCRP_D	SUBSCRP_WEEKLY_CSUS_F

ETL Jobs for Analytics

The analytics jobs are used by the analytical components of SAS Offer Optimization for Communications. These jobs extract data from the Foundation data mart and then load this data in the base tables.

The following table lists the jobs that you have to run in SAS Data Integration Studio for managing data that is required for the analytical components.

Table 3.3 *Analytics Jobs*

Job Name	Purpose	Primary Source Table	Target Table
cfd_pre_pd_cust_interaction_b_job	loads interaction-related data (that is, count of inquiries and complaints for prepaid customers). This job loads data for the completed week.	• CUST_INQUIRY_F • CUST_COMPLAINT_F	PRE_PD_CUST_INTERACTION_B
cfd_pre_pd_cust_loyalty_b_job	loads loyalty-related data such as loyalty points earned and loyalty bonus points earned for prepaid customers. This job loads data for the completed week.	LOYALTY_ACTIVITY_F	PRE_PD_CUST_LOYALTY_B
cfd_pre_pd_equip_activity_b_job	loads information about status and their corresponding change reason for various equipment that is owned by prepaid customers. This job loads data for the completed week.	SUBSCRP_EQUIP_ACTIVITY_F	PRE_PD_EQUIP_ACTIVITY_B
cfd_pre_pd_service_activity_b_job	loads information about various services and their status that are owned by prepaid customers. This job loads data for the completed week.	SUBSCRP_SERVICE_ACTIVITY_F	PRE_PD_SERVICE_ACTIVITY_B
cfd_pre_pd_subscrp_bucket_drvd_b_job	loads various bucket-level customer variables that are precalculated for analytical processing such as calculating the duration between first recharge from activation. This job loads data for the completed week.	• PREPAID_NON_USAGE_CHARGE_F • PREPAID_USAGE_CHARGE_F • SUBSCRP_BUCKET_USAGE_SNPSHT_F	PRE_PD_SUBSCRP_BUCKET_DRVD_B

Job Name	Purpose	Primary Source Table	Target Table
cfd_pre_pd_subscrp_bucket_usage_b_job	loads a daily snapshot of the recharge closing balance and usage decrement value for prepaid customers. This job loads data for the completed week.	SUBSCRP_BUCKET_USAGE_SNPSHT_F	PRE_PD_SUBSCRP_BUCKET_USAGE_B
cfd_pre_pd_subscrp_interaction_b_job	loads interaction information at subscription level for prepaid customers. This job loads data for the completed week.	• CUST_INQUIRY_F • CUST_COMPLAINT_F	PRE_PD_SUBSCRP_INTERACTION_B
cfd_pre_pd_subscrp_loyalty_b_job	loads information about loyalty points for prepaid customers at a subscription level. This job loads data for the completed week.	LOYALTY_ACTIVITY_F	PRE_PD_SUBSCRP_LOYALTY_B
cfd_pre_pd_subscrp_usage_drvd_b_job	loads various precalculated variables at subscription level. These variables are required for analytical processing such as calculating duration of last recharge from first usage. This job loads data for the completed week.	• PREPAID_NON_USAGE_CHARGE_F • PREPAID_USAGE_CHARGE_F	PRE_PD_SUBSCRP_USAGE_DRVD_B
cfd_pre_pd_usage_b_job	loads usage information for prepaid customers. This job loads data for the completed week.	• USAGE_SUMMARY_F • EVENT_FAILURE_SUMMARY_F	PRE_PD_USAGE_B
cfd_pre_pd_usage_recharge_b_job	loads recharge information for prepaid customers. This job loads data for the completed week.	• PAYMENT_F • PREPAID_USAGE_CHARGE_F	PRE_PD_USAGE_RECHARGE_B
cfd_pst_pd_bill_usage_b_job	loads information about billing-usage data for the postpaid customers.	BILL_USAGE_F	PST_PD_BILL_USAGE_B
cfd_pst_pd_cust_acct_snpsht_b_job	loads derived information about customers' accounts for postpaid customers. This job loads data for the completed billing cycle.	CUST_ACCT_BALANCE_SNPSHT_F	PST_PD_CUST_ACCT_SNPSHT_B
cfd_pst_pd_cust_bill_b_job	loads bill-related information for postpaid customers. This job loads data for the completed billing cycle.	BILL_F	PST_PD_CUST_BILL_B

Job Name	Purpose	Primary Source Table	Target Table
cfd_pst_pd_cust_bill_nonusage_b_job	loads bill-non-usage-related information for postpaid customers. This job loads data for the completed billing cycle.	BILL_NON_USAGE_F	PST_PD_CUST_BILL_NONUSAGE_B
cfd_pst_pd_cust_interaction_b_job	loads interaction-related information for postpaid customers. This job loads data for the completed billing cycle.	• CUST_INQUIRY_F • CUST_COMPLAINT_F	PST_PD_CUST_INTERACTION_B
cfd_pst_pd_cust_loyalty_b_job	loads information about loyalty points for postpaid customers. This job loads data for the completed bill cycle.	LOYALTY_ACTIVITY_F	PST_PD_CUST_LOYALTY_B
cfd_pst_pd_equip_activity_b_job	loads information related to the status and their corresponding change reason for various equipment that is owned by postpaid customers. This job loads data for the completed bill cycle.	SUBSCRP_EQUIP_ACTIVITY_F	PST_PD_EQUIP_ACTIVITY_B
cfd_pst_pd_payment_base_b_job	loads payment-related information about postpaid customers. This job loads data for the completed bill cycle.	PAYMENT_F	PST_PD_PAYMENT_B
cfd_pst_pd_payment_drvd_b_job	loads payment-related-derived information about postpaid customers. This job loads data for the completed bill cycle.	• BILL_X_PAYMENT_BRIDGE • BILL_F	PST_PD_PAYMENT_DRVD_B
cfd_pst_pd_service_activity_b_job	loads information about various services and their statuses that are owned by postpaid customers. This job loads data for the completed bill cycle.	SUBSCRP_SERVICE_ACTIVITY_F	PST_PD_SERVICE_ACTIVITY_B
cfd_pst_pd_subscrp_bill_nonusage_b_job	loads bill-non-usage-related information about postpaid customers at subscription level. This job loads data for the completed bill cycle.	BILL_NON_USAGE_F	PST_PD_SUBSCRP_BILL_NONUSAGE_B

Job Name	Purpose	Primary Source Table	Target Table
cfd_pst_pd_subscrp_interaction_b_job	loads subscription-level interaction information about postpaid customers. This job loads data for the completed bill cycle.	• CUST_INQUIRY_F • CUST_COMPLAINT_F	PST_PD_SUBSCRP_INTERACTION_B
cfd_pst_pd_subscrp_loyalty_b_job	loads the loyalty points for subscriptions of postpaid customers. This job loads data for the completed billing cycle.	LOYALTY_ACTIVITY_F	PST_PD_SUBSCRP_LOYALTY_B
cfd_pst_pd_usage_b_job	loads usage-related information about postpaid customers. This job loads data for the completed billing cycle.	• USAGE_SUMMARY_F • EVENT_FAILURE_SUMMARY_F	PST_PD_USAGE_B
cfd_pre_pd_cust_offer_snpsht_b_job	loads the latest details of the all the offers that a prepaid customer owns.	PRE_PD_CUST_TMP,OFFER_D	PRE_PD_CUST_OFFER_SNPSHT_B
cfd_pre_pd_cust_snpsht_b_job	loads the details about prepaid customers. These details are related to current and previous offer bundle and subscription status (count of activation and deactivation)	• PRE_PD_CUST_TMP • OFFER_BUNDLE_D,SUBSCRP_D	PRE_PD_CUST_SNPSHT_B
cfd_pst_pd_cust_offer_snpsht_b_job	loads the latest details of the all the offers that a postpaid customer owns.	• CUST_X_BILL_CYCLE_BRIDGE_TMP • OFFER_D	PST_PD_CUST_OFFER_SNPSHT_B
cfd_pst_pd_cust_snpsht_b_job	loads the details about postpaid customers. These details are related to current and previous offer bundle and subscription status (count of activation and deactivation)	• CUST_X_BILL_CYCLE_BRIDGE_TMP • OFFER_BUNDLE_D • SUBSCRP_D	PST_PD_CUST_SNPSHT_B

Chapter 4

Working With Dynamic ABT

Overview of Dynamic ABT

Dynamic ABT (DABT) is a module of SAS Offer Optimization for Communications that enables you to build analytical base tables. These analytical base tables are required for the core analytical processes such as microsegmentation and offer ranking, which are also steps of the SAS Offer Optimization for Communications workflow. In addition, customer retention and customer segmentation, which are the analytical components of SAS Offer Optimization for Communications, also require DABT for building analytical base tables.

Prerequisites of DABT

Overview

Before you start using DABT, you have to populate the Analytics data mart with appropriate data. In addition, you have to complete tasks such as initializing parameters and deploying a master loop job.

Initialize Parameters

The cfd_dabt_param_declaration_job job initializes the parameters that are required for DABT. This job refers to the CFDN_CONFIG_PARAM table that resides in the DABT_ADM library. The CFDN_CONFIG_PARAM table contains names and values of the global parameters. For details, see "Parameters for DABT" on page 239.

After you run the cfd_dabt_param_declaration_job job, the cfdn_initialization.sas parameter file is generated.

To initialize the parameters:

1. Verify the values of all the parameters in the CFDN_CONFIG_PARAM table and modify the values if necessary.
2. Log on to SAS Data Integration Studio and connect to a profile.
3. On the **Folders** tab, expand **Products** ⇨ **SAS Offer Optimization for Communications** ⇨ **Jobs** ⇨ **Analytics Setup Job Group**.
4. Run the cfd_dabt_param_declaration_job job.
5. Close SAS Data Integration Studio.
6. Go to the **`<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode`** folder.
7. Verify that the cfdn_initialize.sas file is created at the specified location.

Deploy Master Loop Job

To deploy the master loop job:

1. Log on to SAS Data Integration Studio and connect to a profile.
2. On the **Folders** tab, expand **Products** ⇨ **SAS Offer Optimization for Communications** ⇨ **Jobs** ⇨ **Dabt Job Group**.
3. Right-click the MasterLoopBASJob job and on the **Scheduling** menu, select **Deploy**.
4. In the **Deploy a job for scheduling** dialog box, complete the following details:
   a. From the **Batch Server** list, select **`SASApp – SAS DATA Step Batch Server`**.
   b. Click **New** to define the deployment directory.
   c. In the New directory dialog box, in the **Name** box, type *dep_jobs*.

d. In the **Directory** box, type **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASMacro**.

e. Click **OK**.

f. Do not change the default name, **MasterLoopBASJob** that is displayed in the **Deployed Job Name** box.

g. Do not change the default value that is displayed in the **Location** box.

h. Click **OK**.

5. Close SAS Data Integration Studio.
6. Go to the**<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASMacro**.
7. Verify that the MasterLoopBASJob.sas file is created at the specified location.

Populating the Analytics Data Mart

The Analytics data mart is categorized into two areas:

- configuration area
- ABT-specific area

Before you populate data into the Analytics data mart, you must be familiar with the DABT data model.

As a post-installation task, default data is populated into the Analytics data mart. However, in order to populate the Analytics data mart completely, you have to complete certain other tasks.

Download the Analytics Workbooks

In order to populate the Analytics data mart, you have to populate certain workbooks, which are in the .xls format. These workbooks are packaged together in a zip file. You have to download this zip file and extract its contents in a suitable location on your computer.

To extract the analytics workbooks:

1. Type or paste the following URL in the address field of your browser: **http://support.sas.com/documentation/solutions/offeropt/index.html**.
2. Enter the user ID and password.
3. Download the zip file that is displayed for the *SAS Offer Optimization for Communications: Analytics Workbooks*.
4. Extract the contents of the zip file to a suitable location on your computer.
5. Verify that the following workbooks are extracted.
   - ABT_SPECIFIC_CR_postpaid.xls
   - ABT_SPECIFIC_CR_prepaid.xls
   - ABT_SPECIFIC_CS_postpaid.xls
   - ABT_SPECIFIC_CS_prepaid.xls
   - ABT_SPECIFIC_MS_postpaid.xls

- ABT_SPECIFIC_MS_prepaid.xls
- ABT_SPECIFIC_OR_postpaid.xls
- ABT_SPECIFIC_OR_prepaid.xls
- BPP_CONFIGURATION.xls
- OOC_Analytical_Data_Dictionary.xls
- ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls
- ABT_SPECIFIC_CSUSprepaid_CustXOffer.xls
- ABT_SPECIFIC_CSUSpostpaid_SubsXService.xls
- ABT_SPECIFIC_CSUSprepaid_SubsXService.xls

6. In Microsoft Excel, open each of these workbooks and familiarize yourself with the contents.

Make sure that you refer to these workbooks when you are familiarizing yourself with the configuration area and the ABT-specific area of DABT. For details, see "About the BPP_Configuration.xls Workbook" on page 42 and "Populating the ABT-Specific Area of Analytics Data Mart" on page 61.

Populating the Configuration Area of Analytics Data Mart

Overview

The configuration area of the Analytics data mart contains metadata for the base tables from which an ABT is built. Default data that is required for internal processing is populated into the configuration area through an INSERT script. For details, see the post-installation instructions in the *SAS Offer Optimization for Communications: Installation and Configuration Guide*, which is located at **http://support.sas.com/documentation/solutions/offeropt/index.html**. However, you have to enter the solution-specific information in the Analytics data mart. In order to facilitate this data entry, you have to first enter the solution-specific information in the BPP_Configuration.xls workbook. You then have to populate this information in the configuration area of the Analytics data mart by running certain macros.

About the BPP_Configuration.xls Workbook

The BPP_Configuration.xls workbook provides you the complete information that you need for populating the configuration area of the Analytics data mart. Make sure that you have extracted this workbook. For details, see "Download the Analytics Workbooks" on page 41. This workbook contains 16 worksheets. Each worksheet contains a set of columns. The following table gives the purpose of each worksheet and also gives information about the columns in each worksheet. In addition, it also indicates how data is populated from the source table to the target tables. This information will help you enter correct data in the relevant worksheet. Here is the information that is detailed in the table below:

Worksheet name
: name of the worksheet as defined in the BPP_Configuration.xls workbook.

Purpose
: a brief explanation of the worksheet.

Macro name
: the macros that read data from the BPP_Configuration.xls workbook and populate data into the target tables of the Analytics mart.

Worksheet Column Name
: name of the column as it appears in the worksheet.

Input Type
: data type of the worksheet column.

Input Length
: length of the worksheet column.

Target Table Name
: the application data mart table that is populated by the worksheet column.

Target Column Name
: the column of the target table (of the application data mart) that is mapped to the worksheet column.

Target Type
: data type of the target column.

Target Length
: length of the target column.

Mapping Logic
: the mapping logic based on which the target column is populated.

Assumptions
: important notes for populating the target table.

Note: It is recommended that you refer to the BPP_CONFIGURATION.xls workbook when you read through this table. This will help you to understand the default values that are populated in the workbook.

Table 4.1 *Worksheets and Columns of BPP_CONFIGURATION.xls Workbook*

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Worksheet name: Library **Purpose:** To facilitate insertion of records in the LIBRARY_MASTER table. **Macro name:** cfdn_config_tab_pop_src								
LIBRARY_SK	NUM		LIBRARY_MASTER	LIBRARY_SK	NUM		Computed by taking max(LIBRARY_SK) from LIBRARY_MASTER	You must enter a numeric value for this column.
LIBRARY_SHORT_NM	CHAR	40	LIBRARY_MASTER	LIBRARY_REFERENCE	CHAR	40	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
LIBRARY_TYPE_CD	CHAR	3	LIBRARY_MASTER	LIBNAME_STATEMENT	CHAR	3	Direct mapping	
LIBRARY_REFERENCE	CHAR	8	LIBRARY_MASTER	LIBRARY_TYPE_CD	CHAR	8	Direct mapping	
LIBNAME_STATEMENT	CHAR	400	LIBRARY_MASTER	LIBRARY_SHORT_NM	CHAR	400	Direct mapping	
LIBRARY_DESC	CHAR	200	LIBRARY_MASTER	LIBRARY_DESC	CHAR	200	Direct mapping	
NA	CHAR	400	LIBRARY_MASTER	LIBRARY_PATH	CHAR	400	NA	This column is not populated during configuration. Customer retention and customer segmentation will internally use this column.
Worksheet name: Time_Frequency **Purpose:** To facilitate insertion of records in the TIME_FREQUENCY_MASTER table. **Macro name:** cfdn_config_tab_pop_one_time								
TIME_FREQUENCY_SK	NUM		TIME_FREQUENCY_MASTER	TIME_FREQUENCY_SK	NUM		Direct mapping	You must enter a numeric value for this column.
TIME_FREQUENCY_CD	CHAR	3	TIME_FREQUENCY_MASTER	TIME_FREQUENCY_CD	CHAR	3	Direct mapping	
TIME_FREQUENCY_SHORT_NM	CHAR	40	TIME_FREQUENCY_MASTER	TIME_FREQUENCY_SHORT_NM	CHAR	40	Direct mapping	
TIME_FREQUENCY_DESC	CHAR	200	TIME_FREQUENCY_MASTER	TIME_FREQUENCY_DESC	CHAR	200	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Worksheet name: Aggregation_Type **Purpose:** To facilitate insertion of records in the AGGREGATION_TYPE_MASTER table. **Macro name:** cfdn_config_tab_pop_one_time								
AGGREGATION_TYPE_SK	NUM		AGGREGATION_TYPE_MASTER	AGGREGATION_TYPE_SK	NUM		Direct mapping	You must enter a numeric value for this column.
AGGREGATION_TYPE_CD	CHAR	3	AGGREGATION_TYPE_MASTER	AGGREGATION_TYPE_CD	CHAR	3	Direct mapping	
AGGREGATION_TYPE_SHORT_NM	CHAR	40	AGGREGATION_TYPE_MASTER	AGGREGATION_TYPE_SHORT_NM	CHAR	40	Direct mapping	
AGGREGATION_TYPE_DESC	CHAR	200	AGGREGATION_TYPE_MASTER	AGGREGATION_TYPE_DESC	CHAR	200	Direct mapping	
Worksheet name: Purpose **Purpose:** To facilitate insertion of records in the PURPOSE_MASTER table. **Macro name:** cfdn_config_tab_pop_src								
PURPOSE_SK	NUM		PURPOSE_MASTER	PURPOSE_SK	NUM		Computed by taking max(PURPOSE_SK) from PURPOSE_MASTER	You must enter a numeric value for this column.
PURPOSE_CD	CHAR	10	PURPOSE_MASTER	PURPOSE_CD	CHAR	10	Direct mapping	
PURPOSE_SHORT_NM	CHAR	40	PURPOSE_MASTER	PURPOSE_SHORT_NM	CHAR	40	Direct mapping	
PURPOSE_DESC	CHAR	200	PURPOSE_MASTER	PURPOSE_DESC	CHAR	200	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Worksheet name: Levels **Purpose:** To facilitate insertion of records in the SOURCE_LEVEL_MASTER table. **Macro name:** cfdn_config_tab_pop_src								
LEVEL_SK	NUM		SOURCE_LEVEL_MASTER	LEVEL_SK	NUM		Computed by taking max(LEVEL_SK) from SOURCE_LEVEL_MASTER	You must enter a numeric value for this column.
LEVEL_KEY_COLUMN_NM	CHAR	150	SOURCE_LEVEL_MASTER	LEVEL_KEY_COLUMN_NM	CHAR	150	Direct mapping	
LEVEL_SHORT_NM	CHAR	40	SOURCE_LEVEL_MASTER	LEVEL_SHORT_NM	CHAR	40	Direct mapping	
LEVEL_DESC	CHAR	200	SOURCE_LEVEL_MASTER	LEVEL_DESC	CHAR	200	Direct mapping	
DATE_COLUMN_NM_FOR_AVG_CALC	CHAR	30	SOURCE_LEVEL_MASTER	DATE_COLUMN_NM_FOR_AVG_CALC	CHAR	30	Direct mapping	
Worksheet name: Time_Period **Purpose:** To facilitate insertion of records in the SOURCE_TIME_PERIOD table. **Macro name:** cfdn_config_tab_pop_src								
NA	NA		SOURCE_TIME_PERIOD	TIME_PERIOD_SK	NUM		Computed by taking max(TIME_PERIOD_SK) from SOURCE_TIME_PERIOD	Although the target column does not appear in the worksheet, it is populated automatically.
TIME_PERIOD_CD	CHAR	8	SOURCE_TIME_PERIOD	TIME_PERIOD_CD	CHAR	8	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
TIME_FREQUENCY_CD	CHAR	8	SOURCE_TIME_PERIOD	TIME_FREQUENCY_SK	NUM		time_frequency_master.time_frequency_cd matched with SOURCE_TIME_PERIOD.TIME_FREQUENCY_CD to fetch TIME_FREQUENCY_SK from time_frequency_master	The length of TIME_FREQUENCY_CD should not exceed 8 characters. The value that you enter for the TIME_FREQUENCY_CD column must be the same as the value of the SOURCE_TIME_PERIOD.TIME_FREQUENCY_CD column.
TIME_PERIOD_FROM	NUM		SOURCE_TIME_PERIOD	TIME_PERIOD_TO	NUM		Direct mapping	
TIME_PERIOD_TO	NUM		SOURCE_TIME_PERIOD	TIME_PERIOD_FROM	NUM		Direct mapping	
TIME_PERIOD_SHORT_NM	CHAR	40	SOURCE_TIME_PERIOD	TIME_PERIOD_SHORT_NM	CHAR	40	Direct mapping	
TIME_PERIOD_DESC	CHAR	200	SOURCE_TIME_PERIOD	TIME_PERIOD_DESC	CHAR	200	Direct mapping	
Worksheet name: Source_Table_Usage **Purpose:** To facilitate insertion of records into TABLE_USAGE_MASTER table. **Macro name:** cfdn_config_tab_pop_one_time								
TABLE_USAGE_SK	NUM		TABLE_USAGE_MASTER	TABLE_USAGE_SK	NUM		Direct mapping	
TABLE_USAGE_CD	CHAR	3	TABLE_USAGE_MASTER	TABLE_USAGE_CD	CHAR	3	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
TABLE_USAGE_SHORT_NM	CHAR	40	TABLE_USAGE_MASTER	TABLE_USAGE_SHORT_NM	CHAR	40	Direct mapping	
TABLE_USAGE_DESC	CHAR	200	TABLE_USAGE_MASTER	TABLE_USAGE_DESC	CHAR	200	Direct mapping	
Worksheet name: Source_Table **Purpose:** Insert records in the SOURCE_TABLE_MASTER. **Macro name:** cfdn_config_tab_pop_src								
SOURCE_TABLE_SK	NUM		SOURCE_TABLE_MASTER, SOURCE_TABLE_X_USAGE, SOURCE_TABLE_X_LEVEL	SOURCE_TABLE_SK	NUM		Direct mapping	
SOURCE_TABLE_NM	CHAR	30	SOURCE_TABLE_MASTER	SOURCE_TABLE_NM	CHAR	30	Direct mapping	
SOURCE_TABLE_CD	CHAR	3	SOURCE_TABLE_MASTER	SOURCE_TABLE_CD	CHAR	3	Direct mapping	
BEH_VAR_USAGE	CHAR	3	SOURCE_TABLE_X_USAGE	table_usage_sk	NUM		table_usage_master.table_usage_cd and BEH_VAR_USAGE matched with cfdn_code_behavioral to fetch the correct table_usage_sk	Enter *BEH* as the value for this column to indicate usage as source for behavioral variables.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
SPM_VAR_USAGE	CHAR	3	SOURCE_TABLE_X_USAGE	table_usage_sk	NUM		table_usage_master.table_usage_cd and SPM_VAR_USAGE matched with cfdn_code_suppl fetch the correct table_usage_sk	Enter *SPM* as the value for this column to indicate usage as source for supplementary variables.
RNT_VAR_USAGE	CHAR	3	SOURCE_TABLE_X_USAGE	table_usage_sk	NUM		table_usage_master.table_usage_cd and RNT_VAR_USAGE cfdn_code_recent matched with to fetch the correct table_usage_sk	Enter *RNT* as the value for this column to indicate usage as source for recent variables.
SUBSET_USAGE	CHAR	3	SOURCE_TABLE_X_USAGE	table_usage_sk	NUM		table_usage_master.table_usage_cd and SUBSET_USAGE matched with cfdn_code_subsetquery to fetch the correct table_usage_sk	Enter *SOY* as the value for this column to indicate usage as source for subsetting.
SOURCE_TABLE_SHORT_NM	CHAR	40	SOURCE_TABLE_MASTER	SOURCE_TABLE_SHORT_NM	CHAR	40	Direct mapping	
SOURCE_TABLE_DESC	CHAR	200	SOURCE_TABLE_MASTER	SOURCE_TABLE_DESC	CHAR	200	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
LIBRARY_NM	CHAR	40	SOURCE_TABLE_MASTER	LIBRARY_SK	NUM		LIBRARY_NMmatched with library_master.library_short_nm to fetch the correct LIBRARY_SK	The length of LIBRARY_NM should not exceed 40 characters. The value that you enter for LIBRARY_NM should be same as the value of the library_master.library_short_nm column.
TIME_FREQUENCY_NM	CHAR	40	SOURCE_TABLE_MASTER	TIME_FREQUENCY_SK	NUM		TIME_FREQUENCY_NM matched with time_frequency_master.time_frequency_short_nm to fetch the correct TIME_FREQUENCY_SK	The length of TIME_FREQUENCY_NM should not exceed 40 characters. The value that you enter for TIME_FREQUENCY_NM should be same as the value of the time_frequency_master.time_frequency_short_nm column.
LEVEL_NM1	CHAR	40	SOURCE_TABLE_X_LEVEL	LEVEL_SK	NUM		source_level_master.level_short_nm is matched with LEVEL_NM1 to fetch the correct LEVEL_SK	A source table can have up to 4 levels.
LEVEL_NM2	CHAR	40	SOURCE_TABLE_X_LEVEL	LEVEL_SK	NUM		source_level_master.level_short_nm is matched with LEVEL_NM1 to fetch the correct LEVEL_SK	The length of LEVEL_NM2 should not exceed 40 characters. The value that you enter for LEVEL_NM2 should be same as the value of the source_level_master.level_short_nm column.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
LEVEL_NM3	CHAR	40	SOURCE_TABLE_X_LEVEL	LEVEL_SK	NUM		source_level_master.level_short_nm is matched with LEVEL_NM1 to fetch the correct LEVEL_SK	The length of LEVEL_NM3 should not exceed 40 characters. The value that you enter for LEVEL_NM3 should be same as the value of the source_level_master.level_short_nm column.
LEVEL_NM4	CHAR	40	SOURCE_TABLE_X_LEVEL	LEVEL_SK	NUM		source_level_master.level_short_nm is matched with LEVEL_NM1 to fetch the correct LEVEL_SK	The length of LEVEL_NM4 should not exceed 40 characters. The value entered for LEVEL_NM4 should match with source_level_master.level_short_nm
Worksheet name: Source_Column_Usage **Purpose:** To facilitate insertion of records in the Source_Column_Usage table. **Macro name:** cfdn_config_tab_pop_one_time								
COLUMN_USAGE_SK	NUM		COLUMN_USAGE_MASTER	COLUMN_USAGE_SK	NUM		Direct mapping	
COLUMN_USAGE_CD	CHAR	3	COLUMN_USAGE_MASTER	COLUMN_USAGE_CD	CHAR	3	Direct mapping	
COLUMN_USAGE_SHORT_NM	CHAR	40	COLUMN_USAGE_MASTER	COLUMN_USAGE_SHORT_NM	CHAR	40	Direct mapping	
COLUMN_USAGE_DESC	CHAR	200	COLUMN_USAGE_MASTER	COLUMN_USAGE_DESC	CHAR	200	Direct mapping	
Worksheet name: Source_Column_Type **Purpose:** To facilitate insertion of records in the COLUMN_TYPE_MASTER table. **Macro name:** cfdn_config_tab_pop_one_time								

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
COLUMN_TYPE_SK	NUM		COLUMN_TYPE_MASTER	COLUMN_TYPE_SK	NUM		Direct mapping	
COLUMN_TYPE_CD	CHAR	3	COLUMN_TYPE_MASTER	COLUMN_TYPE_CD	CHAR	3	Direct mapping	
COLUMN_TYPE_SHORT_NM	CHAR	40	COLUMN_TYPE_MASTER	COLUMN_TYPE_SHORT_NM	CHAR	40	Direct mapping	
COLUMN_TYPE_DESC	CHAR	200	COLUMN_TYPE_MASTER	COLUMN_TYPE_DESC	CHAR	200	Direct mapping	
Worksheet name: Source_Columns **Purpose:** To facilitate insertion of records in the SOURCE_COLUMN_MASTER table. **Macro name:** cfdn_config_tab_pop_src								
NA	NA		SOURCE_COLUMN_MASTER source_column_x_usage	SOURCE_COLUMN_SK			Computed by taking max(SOURCE_COLUMN_SK) from SOURCE_COLUMN_MASTER	Although the target column does not appear in the worksheet, it is populated automatically.
SOURCE_TABLE_NM	CHAR	30	SOURCE_COLUMN_MASTER	SOURCE_TABLE_SK	NUM		To fetch the correct SOURCE_COLUMN_SK belonging to the correct SOURCE_TABLE that the column belongs to	Source table name should not exceed 30 characters because it will be compared with the physical name of the table that is indicated by source_table_master.source_table_nm.
COLUMN_NM	CHAR	30	SOURCE_COLUMN_MASTER	SOURCE_COLUMN_NM	CHAR	30	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
COLUMN_CD	CHAR	8	SOURCE_COLUMN_MASTER	SOURCE_COLUMN_CD	CHAR	8	Direct mapping	
COLUMN_SHORT_NM	CHAR	40	SOURCE_COLUMN_MASTER	SOURCE_COLUMN_SHORT_NM	CHAR	40	Direct mapping	
COLUMN_DESC	CHAR	200	SOURCE_COLUMN_MASTER	SOURCE_COLUMN_DESC	CHAR	200	Direct mapping	
COLUMN_TYPE	CHAR	3	SOURCE_COLUMN_MASTER	COLUMN_TYPE_SK	NUM		column_type compared with cfdn_code_measures, cfdn_code_dimensions, cfdn_code_keys , cfdn_code_dates and the correct value assigned to column_type_cd. column_type_master.column_type_cd matched with column_type_cd to fetch the correct COLUMN_TYPE_SK	Specify the list of values (DAT, MSR, DIM, and KEY) to indicate date, measure, and dimension key type of the source column.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
IS_RECENT_DT	CHAR	8	source_column_x_usage	column_usage_sk	NUM		is_recent_dt and column_usage_master.column_usage_cd are compared with cfdn_code_recent_dt to fetch the correct column_usage_master.column_usage_sk	Enter *RNT* to indicate recent as the column usage for date type of column. This indicates that the specified date would be used to retrieve the latest record.
IS_EXTRACTION_DT	CHAR	8	source_column_x_usage	column_usage_sk	NUM		is_extraction_dt and column_usage_master.column_usage_cd are compared with cfdn_code_extraction_dt to fetch the correct column_usage_master.column_usage_sk	Enter *DT_EXTR* to indicate usage as date of extraction for the date type of column. This indicates that the specified date would be used to sort the records while building behavioral variables.
IS_VALID_FROM_DT	CHAR	8	source_column_x_usage	column_usage_sk	NUM		is_valid_from_dt and column_usage_master.column_usage_cd are compared with cfdn_code_validfr_dt to fetch the correct column_usage_master.column_usage_sk	Enter *VALID_FR* to indicate usage as valid from date for the date type of columns. This would be used for extracting valid values from dimension-like table while building supplementary variables

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
IS_VALID_TO_DT	CHAR	8	source_column_x_usage	column_usage_sk	NUM		is_valid_to_dt and column_usage_master.column_usage_cd are compared with cfdn_code_validto_dt to fetch the correct column_usage_master.column_usage_sk	Enter *VALID_TO* indicate usage as valid to date for the date type of columns. This would be used for extracting valid values from dimension-like table while building supplementary variables
Worksheet name: Dim_Attribute_Values **Purpose:** To facilitate insertion of records in the SOURCE_DIM_ATTRIB_VALUE_MASTER table. **Macro name:** cfdn_config_tab_pop_dimval								
NA	NUM		SOURCE_DIM_ATTRIB_VALUE_MASTER	DIM_ATTRIBUTE_VALUE_SK	NUM		Computed by taking max(DIM_ATTRIBUTE_VALUE_SK) from SOURCE_COLUMN_MASTER	Although the target column does not appear in the worksheet, it is populated automatically
SOURCE_COLUMN_NM	CHAR	30	SOURCE_DIM_ATTRIB_VALUE_MASTER	SOURCE_COLUMN_SK	NUM		SOURCE_COLUMN_NM is matched with source_column_master.source_column_nm to fetch the correct SOURCE_COLUMN_SK	Source column name should not exceed 30 characters because it will be compared with the physical name of the column indicated by source_column_master.source_column_nm

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
SOURCE_TABLE_NM	CHAR	30	SOURCE_DIM_ATTRIB_VALUE_MASTER	NA	NA		Used to pick out the correct SOURCE_COLUMN_SK belonging to SOURCE_TABLE_NM	Column names can repeat across tables. Therefore, you must specify the table name to which the column belongs.
DIM_ATTRIBUTE_VALUE	CHAR	40	SOURCE_DIM_ATTRIB_VALUE_MASTER	DIM_ATTRIBUTE_VALUE	CHAR	40	Direct mapping	
DIM_ATTRIBUTE_VALUE_CD	CHAR	3	SOURCE_DIM_ATTRIB_VALUE_MASTER	DIM_ATTRIBUTE_VALUE_CD	CHAR	3	Direct mapping	
DIM_ATTRIBUTE_VALUE_SHORT_NM	CHAR	40	SOURCE_DIM_ATTRIB_VALUE_MASTER	DIM_ATTRIBUTE_VALUE_SHORT_NM	CHAR	40	Direct mapping	
DIM_ATTRIBUTE_VALUE_DESC	CHAR	200	SOURCE_DIM_ATTRIB_VALUE_MASTER	DIM_ATTRIBUTE_VALUE_DESC	CHAR	200	Direct mapping	
Worksheet name: Variable_Type **Purpose:** To facilitate insertion of records in the VARIABLE_TYPE_MASTER table. **Macro name:** cfdn_config_tab_pop_one_time								
VARIABLE_TYPE_SK	NUM		VARIABLE_TYPE_MASTER	VARIABLE_TYPE_SK	NUM		Direct mapping	
VARIABLE_TYPE_CD	CHAR	3	VARIABLE_TYPE_MASTER	VARIABLE_TYPE_CD	CHAR	3	Direct mapping	
VARIABLE_TYPE_SHORT_NM	CHAR	40	VARIABLE_TYPE_MASTER	VARIABLE_TYPE_SHORT_NM	CHAR	40	Direct mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
VARIABLE_TYPE_DESC	CHAR	200	VARIABLE_TYPE_MASTER	VARIABLE_TYPE_DESC	CHAR	200	Direct mapping	
Worksheet name: Subset_From_Path **Purpose:** To facilitate insertion of records in the SUBSET_FROM_PATH_MASTER table. **Macro name:** cfdn_config_tab_pop_subset								
SUBSET_FROM_PATH_SK	NUM		SUBSET_FROM_PATH_MASTER	SUBSET_FROM_PATH_SK	NUM		Computed by taking max(SUBSET_FROM_PATH_SK) from SUBSET_FROM_PATH_MASTER	You must enter a numeric value for this column.
FROM_PATH_NM	CHAR	40	SUBSET_FROM_PATH_MASTER	FROM_PATH_SHORT_NM	CHAR	40	Direct mapping	
FROM_PATH_DESC	CHAR	200	SUBSET_FROM_PATH_MASTER	FROM_PATH_DESC	CHAR	200	Direct mapping	
Worksheet name: From_Path_Level **Purpose:** To facilitate insertion of records in the SUBSET_FROM_PATH_X_LEVEL table. **Macro name:** cfdn_config_tab_pop_subset								
FROM_PATH_NM	CHAR	40	SUBSET_FROM_PATH_X_LEVEL	SUBSET_FROM_PATH_SK	NUM		FROM_PATH_NM is matched with subset_from_path_master.from_path_short_nm to fetch the appropriate SUBSET_FROM_PATH_SK	FROM_PATH_NM should not exceed 40 characters because it will be compared with the subset_from_path_master.from_path_short_nm column.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
LEVEL_NM	CHAR	40	SUBSET_FROM_PATH_X_LEVEL	LEVEL_SK	NUM		LEVEL_NM is matched with source_level_master.level_short_nm to fetch the correctLEVEL_SK	LEVEL_NM should not exceed 40 characters because it will be compared with source_level_master.level_short_nm.
SELECT_SOURCE_TABLE_COLUMN_NM	CHAR	150	SUBSET_FROM_PATH_X_LEVEL	SELECT_SOURCE_TABLE_COLUMN_NM	CHAR	150	Direct mapping	
Worksheet name: Subset_Join_Condition **Purpose:** To facilitate insertion of records in the SUBSET_TABLE_JOIN_CONDITION table. **Macro name:** cfdn_config_tab_pop_subset								
SUBSET_TABLE_JOIN_CONDITION_SK	NUM		SUBSET_TABLE_JOIN_CONDITION	SUBSET_TABLE_JOIN_CONDITION_SK	NUM		Direct mapping	
JOIN_CONDITION_SEQUENCE_NUMBER	NUM		SUBSET_TABLE_JOIN_CONDITION	JOIN_CONDITION_SEQUENCE_NUMBER	NUM		Direct mapping	
FROM_PATH_NM	CHAR	40	SUBSET_TABLE_JOIN_CONDITION	SUBSET_FROM_PATH_SK	NUM		FROM_PATH_NM is matched with subset_from_path_master.from_path_short_nm to fetch the appropriate SUBSET_FROM_PATH_SK	FROM_PATH_NM should not exceed 40 characters because it will be compared with subset_from_path_master.from_path_short_nm.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
JOIN_TYPE	CHAR	8	SUBSET_TABLE_JOIN_CONDITION	JOIN_TYPE	CHAR	8	Direct mapping	You can enter any one of the following values: NONE, RIGHT, LEFT, and INNER.
LEFT_SOURCE_TABLE_NM	CHAR	40	SUBSET_TABLE_JOIN_CONDITION	LEFT_TABLE_SK	NUM		LEFT_SOURCE_TABLE_NM is compared with source_table_master.source_table_nm to fetch the correct SOURCE_TABLE_SK	LEFT_SOURCE_TABLE_NM should not exceed 30 characters because it will be compared with the physical name of the table that is indicated by source_table_master.source_table_nm.
LEFT_SOURCE_COLUMN_NM	CHAR	30	SUBSET_TABLE_JOIN_CONDITION	LEFT_COLUMN_SK	NUM		LEFT_SOURCE_COLUMN_NM is compared with source_column_master.source_column_nm to fetch the correct SOURCE_COLUMN_SK	LEFT_SOURCE_COLUMN_NM should not exceed 30 characters because it is will be compared with physical name of the column that is indicated by source_column_master.source_column_nm.
RIGHT_SOURCE_TABLE_NM	CHAR	30	SUBSET_TABLE_JOIN_CONDITION	RIGHT_TABLE_SK	NUM		LEFT_SOURCE_TABLE_NM is compared with source_table_master.source_table_nm to fetch the correct SOURCE_TABLE_SK	For join type NONE, right table SK is not used because there is only one table participating in the subset besides the base tables. RIGHT_SOURCE_TABLE_NM should not exceed 30 characters because it will be compared with physical name of the table indicated by source_table_master.source_table_nm.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
RIGHT_SOURCE_COLUMN_NM	CHAR	30	SUBSET_TABLE_JOIN_CONDITION	RIGHT_COLUMN_SK	NUM		RIGHT_SOURCE_COLUMN_NM is compared with source_column_master.source_column_nm to fetch the correct SOURCE_COLUMN_SK	For join type NONE, right column SK is not used because there is only one table participating in the subset besides the base tables. RIGHT_SOURCE_COLUMN_NM should not exceed 30 characters because it will be compared with the physical name of the column that is indicated by source_column_master.source_column_nm.

Working with the BPP_Configuration.xls Workbook

Data is prepopulated in the Analytics data mart through INSERT scripts. DABT requires this data for internal processing. Therefore, you must not update this data and also must not change the information that is already populated in the following worksheets of the BPP_Configuration.xls workbook.

- Time_Frequency
- Aggregation_Type
- Purpose
- Levels
- Source_Table_Usage
- Source_Column_Usage
- Source_Type_Usage
- Variable_Type

However, you have to manually populate the configuration area every time you add a base table or a column in a table. To do so, you have to enter the correct data in the following worksheets of the BPP_CONFIGURATION.xls workbook:

- Source_Table
- Source_Columns

In addition, verify and modify the data that is populated in the following worksheets:

Library
: Verify the values of library paths and library statements before running the macros that populate the configuration area of the Analytics data mart.

Time_Period
: If the analytical variables cover history in weeks or months beyond what is already specified in this worksheet, then make the required changes and run the appropriate macro.

Dim_Attribute_Values
: The names and values of analytical variables that are defined in the ABT use the values of the dimensional attributes. Because these variables are implementation-specific, they cannot be prepopulated. Therefore, the values of dimension attributes that are mentioned in the source_column_master table must be specified in the **Dim_Attribute_Values** worksheet. Use the specified macro to enter values in the Analytics data mart.

Subset_From_Path, From_Path_Level, and Subset_Join_Condition
: You can use the subsetting feature as a part of ABT building. This feature can be used for customer retention and customer segmentation modules. You have to first decide on the subset criteria, and then proceed to work with the following three worksheets:

- Subset_from_path
- From_Path_Level
- Subset_Join_Condition

Populating the ABT-Specific Area of Analytics Data Mart

Overview

The ABT-specific area of the Analytics data mart stores information about the structure of ABTs that are to be created. The definition of an ABT also refers to the descriptions of base tables that are included in the configuration area.

Before building ABTs, you must specify the structure of the ABT along with the details of the variables that it contains. You have to specify these details in the ABT-specific workbook. You have to then populate this information in the ABT-specifc area of the Analytics data mart by running certain macros.

About the ABT-Specific Workbook

The ABT-specific workbook provides you with the complete information that you need for populating the ABT-specific area of the Analytics data mart. A separate workbook is created depending on the purpose for which the ABT is being built. Make sure that you have downloaded these workbooks. For details, see "Download the Analytics Workbooks" on page 41. Each workbook contains 7 worksheets. Each worksheet contains a set of columns. The following table gives the purpose of each worksheet and also gives information about the columns of each worksheet. In addition, it indicates how data is populated from the source table to the target tables. This information will help you enter correct data in the relevant worksheets. Here is the information that is detailed in the table below:

Worksheet name
: name of the worksheet as defined in the ABT-specific workbook.

Purpose
a brief explanation of the worksheet.

Macro sequence
the sequence in which you have to run the macros to populate data into the target table.

Worksheet Column Name
name of the column as it appears in the worksheet.

Input Type
data type of the worksheet column.

Input Length
length of the worksheet column.

Target Table Name
the application data mart table that is populated by the worksheet column.

Target Column Name
the column of the target table (of the application data mart) that is mapped to the worksheet column.

Target Type
data type of the target column.

Target Length
length of the target column.

Mapping Logic
the mapping logic based on which the target column is populated.

Assumptions
important notes for populating the target table.

Note: It is recommended that you refer to any one of the ABT-specific workbooks when you read through this table. This will help you to understand the default values that are populated in the workbook.

Table 4.2 *Worksheets and Columns of ABT-Specific Workbooks*

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Worksheet name: Subset_where **Purpose:** To facilitate creation of a subset query. **Macro sequence:** cfdn_create_subset_wrapper								
NA		NA	SUBSET_QUERY_MASTER	SUBSET_QUERY_SK		NA	Computed by taking max(SUBSET_QUERY_SK) from SUBSET_QUERY_MASTER	This column value is generated automatically in the wrapper codes.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Query_Nm	CHAR	40	SUBSET_QUERY_MASTER	SUBSET_QUERY_SHORT_NM	CHAR	40	Direct Mapping	
Description	CHAR	200	SUBSET_QUERY_MASTER	SUBSET_QUERY_DESC	CHAR	200	Direct Mapping	
WHERE_CLAUSE_USER_EXPRSN_FLG	CHAR	1	SUBSET_QUERY_MASTER	WHERE_CLAUSE_USER_EXPRSN_FLG	CHAR	1	Direct Mapping	This column is not used by DABT. You can keep this column blank.
WHERE_CLAUSE_EXPRESSION	CHAR	400	SUBSET_QUERY_MASTER	WHERE_CLAUSE_EXPRESSION	CHAR	400	Direct Mapping	
from_path_nm	NUM		SUBSET_QUERY_MASTER	SUBSET_FROM_PATH_SK	NUM		Derived by looking up SUBSET_FROM_PATH_SK corresponding to from_path_short_nm	
Worksheet name: project **Purpose:** To facilitate creation of a new project. **Macro sequence:** cfdn_new_project_wrapper.sas								
NA	NUM		PROJECT_MASTER	PROJECT_SK	NUM		Computed by taking max(PROJECT_SK) from PROJECT_MASTER	
PROJECT_NM	CHAR	40	PROJECT_MASTER	PROJECT_SHORT_NM	CHAR	40	Direct Mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
LEVEL_NM	CHAR	40	PROJECT_MASTER	LEVEL_SK	CHAR	40	Compare LEVEL_NM to SOURCE_LEVEL_MASTER.level_short_nm to fetch the correct LEVEL_SK	
DESCRIPTION	CHAR	200	PROJECT_MASTER	PROJECT_DESC	CHAR	200	Direct Mapping	
LOCATION_PATH	NA	NA	PROJECT_MASTER	NA	NA	NA	NA	
Query_Nm	CHAR	40	PROJECT_MASTER	SUBSET_QUERY_SK	CHAR	40	Compare Query_Nm to subset_query_master.subset_query_short_nm to fetch the correct SUBSET_QUERY_SK	
Purpose	NA	NA	PROJECT_MASTER	NA	NA	NA	NA	
Worksheet name: ABT **Purpose:** To facilitate creation of a new ABT. **Macro sequence:** cfdn_new_project_wrapper.sas								
NA	NUM		ABT_MASTER	ABT_SK	NUM		Computed by taking max(PROJECT_SK) from PROJECT_MASTER	
Table_Nm	CHAR	30	ABT_MASTER	ABT_TABLE_NM	CHAR	30	Direct Mapping	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Short_Nm	CHAR	40	ABT_MASTER	ABT_SHORT_NM	CHAR	40	Direct Mapping	
Description	CHAR	200	ABT_MASTER	ABT_DESC	CHAR	200	Direct Mapping	
Target_Time_Freq	CHAR	40	ABT_MASTER	TARGET_PERIOD_TIME_FREQ_SK	NUM		Compare Target_Time_Freq to time_frequency_master.time_frequency_short_nm to fetch the correct TARGET_PERIOD_TIME_FREQ_SK	
Target_Time_Freq_Count	NUM		ABT_MASTER	TARGET_PERIOD_CNT	NUM		Direct Mapping	
Library	CHAR	8	ABT_MASTER	LIBRARY_SK	NUM		Compare Library to library_master.LIBRARY_REFERENCE to fetch the correct LIBRARY_SK	For churn and segmentation, the library of the ABT must point to the library of type TMP. For SAS Offer Optimization for Communications, it must point to a library with name abtdata.
Level	CHAR	40	ABT_MASTER	LEVEL_SK	NUM		Compare Level to source_level_master.level_short_nm to fetch the correct LEVEL_SK	

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
PROJECT_NM	CHAR	40	ABT_MASTER	PROJECT_SK	NUM		Compare PROJECT_NM to project_master.project_short_nm to fetch the correct PROJECT_SK	
FLG	CHAR	1	ABT_MASTER	NA	CHAR	1	NA	Type *C* to indicate creation of ABT and *D* to indicate deletion of ABT.
Purpose	CHAR	10	ABT_MASTER	PURPOSE_SK		10	Compare Purpose to purpose_master.purpose_cd to fetch the correct PURPOSE_SK	
ABT_TYPE	CHAR	1	ABT_MASTER	ABT_TYPE_FLG		1	C for creating ABT and D for deleting the ABT	Enter *M* (modeling) for churn and segmentation; whereas enter *T*(Template) for ABTs that are created for Microsegmentation and Offer Ranking.
Worksheet name: supplementary **Purpose:** To facilitate defining variables of supplementary type and writing the definitions to the Application data mart. **Macro sequence:** • cfdn_abtvars_import • cfdn_create_spm_var_wrapper • cfdn_save_spm_var_wrapper • cfdn_generate_list								
Var_Group	NUM		NA	NA	NA		NA	Variable group is created for facilitating processing of multiple variables of a particular type at one time. While creating the variable definitions, the variables will be sorted groupwise and processed one-by-one.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
ABT_Nm	CHAR	30	NA	ABT_SK	NUM		abt_master.abt_table_nm is matched with ABT_Nm to fetch the correct ABT_SK	The length of ABT_Nm should not exceed 30 characters. The value that you enter for the ABT_Nm column should be the same as the value of the abt_master.abt_table_nm column.
Source_Table_Nm	CHAR	30	NA	SOURCE_TABLE_SK	NUM		source_table_master.source_table_nm is matched with Source_Table_Nm to fetch the correct SOURCE_TABLE_SK	The length of Source_Table_Nm should not exceed 30 characters. The value that you enter for Source_Table_Nm should be the same as the value of the source_table_master.source_table_nm column.
Select_Column_Nm	CHAR	30	"variable_master supplementary_variable abt_x_variable"	SELECT_SOURCE_COLUMN_SK	NUM		source_column_master.source_column_nm is matched with Select_Column_Nm to fetch the correct SOURCE_COLUMN_SK	The length of Select_Column_Nm should not exceed 30 characters. The value that you enter for Select_Column_Nm should be the same as the value of the source_column_master.source_column_nm column. For column type (DIM, MSR, and DAT), Select_Column_Nm represents the physical column of type dimension, measure, and date.
Select_Column_Type_Cd	CHAR	3	NA	NA	NA		Used to determine what type of lists (measure, dimension, time period, aggregation, dimension values, and so on) to be constructed to be passed as parameters to DABT code	The possible values for this column are MSR, DIM, AGG, and TMP. Depending on which variable the variable group represents, the Select_Column_Type_Cd and Select_Column_Nm are used together as the variable group is processed internally. For supplementary variable, either dimension, measure, or date can be used.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Worksheet name: recent **Purpose:** To facilitate defining variables of recent type and writing the definitions to the Application data mart. **Macro sequence:** • cfdn_abtvars_import • cfdn_abt_dimval_insert • cfdn_create_rnt_var_wrapper • cfdn_save_rnt_var_wrapper								
Var_Group	NUM		NA	NA	NA		NA	Variable group is created for processing multiple variables of a particular type together. While creating the variable definitions, the variables will be sorted groupwise and processed one-by-one.
ABT_Nm	CHAR	30				30	abt_master.abt_table_nm is matched with ABT_Nm to fetch the correct ABT_SK	The length of ABT_Nm should not exceed 30 characters. The value that you enter for the ABT_Nm column should be the same as the value of the abt_master.abt_table_nm column.
Source_Table_Nm	CHAR	30				30	source_table_master.source_table_nm is matched with Source_Table_Nm to fetch the correct SOURCE_TABLE_SK	The length of Source_Table_Nm should not exceed 30 characters. The value entered for Source_Table_Nm should be the same as the value of the source_table_master.source_table_nm column.
Select_Column_Nm	CHAR	30	"variable_master recent_variable abt_x_variable"	SELECT_SOURCE_COLUMN_SK		30	source_column_master.source_column_nm is matched with Select_Column_Nm to fetch the correct SOURCE_COLUMN_SK	The length of Select_Column_Nm should not exceed 30 characters. The value that you enter for Select_Column_Nm should be the same as the value fo the source_column_master.source_column_nm column. For column types such as MSR, DIM, and DAT, Select_Column_Nm represents the physical column of type measure, dimension, or date.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Select_Column_Type_Cd	CHAR	3	"variable_master recent_variable abt_x_variable"			3	Used to determine what type of lists (measure, dimension, time period, aggregation, dimension values, and so on) to be constructed to be passed as parameters to DABT code	The possible values for this column are MSR, DIM, or DAT. Depending on which variable the variable group represents, the Select_Column_Type_Cd and Select_Column_Nm are used together as the variable group is processed internally. For recent variable, either dimension, measure, or date can be used.
Select_Column_Value	CHAR	1000	variable_dim_attribute_filter			1000	Used to populate list of values of dimensional attributes	Select column value is applicable to columns of type dimension. This column should be populated for Select_Column_Type_Cd equal to DIM, although Select_Column_Value is optional for defining a recent variable. For multiple dimensional value, use # as the separator. For example, for service SMS and MMS, type the column values as *SMS#MMS*
Order_By_Date	CHAR	30		ORDER_BY_DATE_SOURCE_COLUMN_SK		30	Order_By_Date matched with SOURCE_COLUMN_MASTER.SOURCE_COLUMN_NM to fetch the order by date SK.	Order by date is compulsory for recent variables. This date is used to sort the records in the source table with reference to ABT build date to select the most recent records.

Worksheet name: beh

Purpose: To facilitate creation of variables of behavioral type.

Macro sequence:

- cfdn_abtvars_import
- cfdn_abt_dimval_insert
- cfdn_create_beh_var_wrapper
- cfdn_save_beh_var_wrapper

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Var_Group	CHAR		NA	NA	NA		NA	Variable group is created for processing multiple variables of a particular type together. While creating the variable definitions, the variables will be sorted groupwise and processed one-by-one. You must enter a numeric value in this field.
ABT_Nm	CHAR	30	abt_x_variable	AGGREGATION_TYPE_SK		30	abt_master.abt_table_nm is matched with ABT_Nm to fetch the correct ABT_SK	The length of ABT_Nm should not exceed 30 characters. The value that you enter for the ABT_Nm column should be the same as the value of the abt_master.abt_table_nm column.
Source_Table_Nm	CHAR	30		TIME_PERIOD_SK		30	source_table_master.source_table_nm is matched with Source_Table_Nm to fetch the correct SOURCE_TABLE_SK	The length of Source_Table_Nm should not exceed 30 characters. The value that you enter for Source_Table_Nm should be the same as the value of the source_table_master.source_table_nm column.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Select_Column_Nm	CHAR	30	"variable_master behavioral_variable abt_x_variable"				source_column_master.source_column_nm is matched with Select_Column_Nm to fetch the correct SOURCE_COLUMN_SK	The length of Select_Column_Nm should not exceed 30 characters. The value that you enter for Select_Column_Nm should be the same as the value of the source_column_master.source_column_nm column. For column types such as MSR, DIM, and DAT, Select_Column_Nm represents the physical column of type measure, dimension, or date. For the TMP column type, Select_Column_Nm represents the time_period_cd from source_time_period table. For the AGG column type, Select_Column_Nm represents the aggregation_type_cd from aggregation_master table. When the aggregation function to be used is CNT as in count, note that DABT does not use the DISTINCT clause while using count as the aggregation function.
Select_Column_Type_Cd	CHAR	3	"variable_master behavioral_variable abt_x_variable"			3	Used to determine what type of lists (measure, dimension, time period, aggregation, dimension values, and so on) to be constructed to be passed as parameters to DABT code	The possible values for this column are MSR, DIM, AGG, and TMP. Depending on what variable the variable group represents, the Select_Column_Type_Cd and Select_Column_Nm are used together as the variable group is processed internally. For behavioral variables, specifying a measure type of column is mandatory. Defining a time period (TMP) as well as aggregation (AGG) is also mandatory.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Select_Column_Value	CHAR	1000	variable_dim_attribute_filter			1000	Used to populate list of values of dimensional attributes	Select column value is applicable to columns of type dimension. That means that this column should be populated for the DIM Select_Column_Type_Cd, although Select_Column_Value is optional for defining a behavioral variable. The values should be separated by #. Also, these values must be configured in dimension attribute values.
Worksheet name: derivedvardetails **Purpose:** To facilitate creation of defining derived variables. **Macro sequence:** • cfdn_drvdvars_import • cfdn_save_drvd_var_wrapper								
ABT_Nm	CHAR	30	ABT_X_VARIABLE	ABT_SK		30	abt_master.abt_table_nm is matched with ABT_Nm to fetch the correct ABT_SK	The length of ABT_Nm should not exceed 30 characters. The value that you enter for the ABT_Nm column should be the same as the value of the abt_master.abt_table_nm column.
Derived_Var_Nm	CHAR	28	"DERIVED_VARIABLE DERIVED_VAR_X_EXPRESSION_VAR VARIABLE_MASTER ABT_X_VARIABLE"	VARIABLE_SUB_TYPE_CD		28		The derived variable name should not be same as the name of an existing variable.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Derived_Var_short_Nm	CHAR	40	"DERIVED_VARIABLE DERIVED_VAR_X_EXPRESSION_VAR VARIABLE_MASTER ABT_X_VARIABLE"			40		
Derived_Var_Desc	CHAR	200	"DERIVED_VARIABLE DERIVED_VAR_X_EXPRESSION_VAR VARIABLE_MASTER ABT_X_VARIABLE"			200		
Derived_Var_Flg	CHAR	1	DERIVED_VARIABLE			1		The supplied values can be S for simple variables or C for complex variables. Simple variables are derived variables that are created from existing ABT variables. Complex variables contain one or more simple derived variables.
Expression	CHAR	400	DERIVED_VARIABLE	EXPRESSION_VARIABLE_SK		400		The derived variable will be generated using the expression that you enter in this column. The expression must contain the ABT variable that would be used to generate the derived variable. For details, refer to the assumptions that are added for the churn flag column in the **Churn** worksheet.

Worksheet Column Name	Input Type	Input Length	Target Table Name	Target Column Name	Target Type	Target Length	Mapping Logic	Assumptions
Worksheet name: derivedvarsource **Purpose:** To facilitate creation of derived variables. **Macro sequence:** • cfdn_drvdvars_import • cfdn_save_drvd_var_wrapper								
ABT_Nm	CHAR	30	ABT_X_VARIABLE	ABT_SK		30	abt_master.abt_table_nm is matched with ABT_Nm to fetch the correct ABT_SK	The length of ABT_Nm should not exceed 30 characters. The value that you enter for the ABT_Nm column should be the same as the value of the abt_master.abt_table_nm column.
Derived_Var_Nm	CHAR	28	DERIVED_VARIABLE DERIVED_VAR_X_EXPRESSION_VAR VARIABLE_MASTER			28		The value for this column must be the same as the value of the Derived_Var_Nm column described in the derivedvardetails worksheet.
ABT_Var_Nm	CHAR	30	DERIVED_VARIABLE DERIVED_VAR_X_EXPRESSION_VAR VARIABLE_MASTER			30		Enter the name of the ABT variable that is used in generating the derived variable. The ABT variable must the abt_name that is specified for the derived variable. This column indicates the ABT variable from which the derived variable is created.

Instructions for populating the correct information in the workbook are given in the relevant sections of *Chapter 6 Customer Retention and Customer Segmentation* of this guide.

Assumptions for Populating Data into Analytics Data Mart

Data is populated into the Analytics data mart based on the following assumptions:

- As a post-installation activity, default data is populated into the Analytics data mart. Any deviation from these values will produce erroneous results. Therefore, you must not change this data.
- Errors can occur if data with incorrect lengths and data types is specified in the worksheets. The DABT module does not handle these errors.
- Activities such as modifying or deleting variables and modifying or deleting data in the configuration area are not supported. Therefore, if you perform any of these tasks, they might produce erroneous results.

  Note: If you want to clean data by modifying or deleting the prepopulated values, then you have to do it manually.
- Analytical base tables (ABTs) are created in the SAS environment. The code does not support ABT creation on any other platform and technology.
- When you populate information about a variable or an ABT into the Analytics data mart through the workbooks, make sure that you comply with the following rules. Otherwise, this might produce erroneous results.

Table 4.3 *Rules for Populating Variables*

	Behavioral	Recent	Supplementary
Measures	Mandatory	Any one of the three	Any one of the three
Dimensions	Not applicable		
Dates	Not applicable		
Dimension values	Optional	Optional	Not applicable
Order by date	Not applicable	Mandatory	Not applicable
Aggregation	Mandatory	Not applicable	Not applicable
Time period	Mandatory	Not applicable	Not applicable

Error Logging for DABT

Errors can occur when you run macros to populate the configuration area and ABT-specific area of the Analytics data mart. These errors are handled through cfdn_rpt_err and cfdn_add_err_cd macros. When an error occurs, it is logged in the cfdn_step_error data set that resides in ERR_LIB SAS library. This library points to the **<SAS configuration path>/Lev1/Applications/SASFoundforComm5.2/Data/dabt_data/error** location. Also, the message that is displayed in the SAS log if an error occurs is localized. The localization is according to the text that is mentioned in the SMD data sets that are located in sashelp folder. For details, see "Configuring Error Tables for Workflow Steps" on page 203.

Similarly, logs are generated when you run macros to populate the configuration area and ABT-specific area of the Analytics data mart. These logs are stored in the **<SAS configuration path>/Lev1/Applications/SASFoundforComm5.2/Logs/**

dabt_logs/flex/<date of execution> path. For example, if you run the macros on July 4, 2010, then the logs are generated in the **<SAS configuration path>/Lev1/Applications/SASFoundforComm5.2/Logs/dabt_logs/flex/04Jun10** location.

Chapter 5

Introduction to Customer Retention and Customer Segmentation

About Customer Segmentation and Customer Retention

Overview

Customer retention and customer segmentation are the analytical components of SAS Offer Optimization for Communications.

Customer Retention
: Using appropriate analytical techniques, derive the probability that customers will leave a communications service provider and avail themselves of a competitor's offer because of not being satisfied with the current offer. Churn scores are produced as an output of this component.

Customer Segmentation
: Using appropriate analytical techniques, divide the population of a communications service provider into homogeneous clusters. Based on these clusters, you can define

the behavior of customers and target customers for marketing campaigns. Analytical segments are the output of this component.

Depending on the underlying business problem, business analysts can determine whether they should choose the churn score or the analytical segment while defining the target segment.

Dependency on Business Groups

As a user of SAS Offer Optimization for Communications, business analysts create projects for each business group. Each business group is an input for customer retention and customer segmentation models. As a result, a data set that contains all modeling variables for every single business group forms the ABT for customer retention and customer segmentation models. Therefore, you can work on these analytical components only after you have defined business groups. For details about defining a business group, see the *SAS Offer Optimization for Communications: User's Guide*.

Assumptions

Here are some of the assumptions that SAS Offer Optimization for Communications makes about business groups and analytical models.

- A DABT project builds a solution-specific ABT. Therefore, a DABT project caters to only one modeling ABT.
- A subset criterion is associated with only one modeling ABT and therefore with only one project.
- There will be different subsetting queries for modeling and scoring ABTs.
- You have to specify the name of the modeling ABT in the ABT-specific workbook. Corresponding scoring ABT names will be generated with **`an _SCR`** suffix.
- You have to run the build_abt macro for every new ABT that is to be built with a specified build date associated with it.

Processing Modes

Customer retention and customer segmentation processes are supported in two modes:

Model Building
: In model building mode, the analytical user considers a sample of the entire communications service provider's data and builds an analytical model on the sample data. The model learns through different behavioral patterns in the sample data. After the user is satisfied with the results that the model produces, it is then registered in the metadata and used in the scoring mode. The sample that is drawn for building a model in the modeling run has to be representative of data that is required in the scoring run.

Scoring or Batch Run
: In the scoring mode, you have to run certain ETL jobs in order to generate scores. Tasks such as ABT building and generating scores and writing them to the Foundation data mart are performed periodically (either monthly or weekly). Most of the configurations that the analytical user defines in model building mode are used in scoring mode.

About the Model Building Mode

Overview

When an analytical model is built, the following aspects are considered:

- the level and time grain at which the analytical model needs to be built
- the historical data that it is required
- the frequency at which customers are scored
- the time window definitions that consider the history and frequency together

Modeling Level

The model building mode assumes that customers are the decision makers who can either accept or reject the offers that they receive.

A customer segmentation model is built at customer level. If the data is available at subscription level, then the DABT processes aggregate the corresponding variables to customer level.

Conversely, customer retention is exhibited by a subscription and not a customer. Therefore, a customer retention model is built at subscription level. The variables that are defined at customer level are repeated for all subscriptions of a customer. The modeling or scoring process generates churn propensity scores for every subscription in the ABT. SAS Offer Optimization for Communications targets to retain customers by offering them best offers. Therefore, churn scores need to be computed at customer level. This can be achieved by aggregating scores from subscription level to customer level. Scores can be aggregated by using weighted means.

Historical Data

Customer retention considers monthly historical data for postpaid customers and requires at least 6 months of data. Conversely, this component considers weekly data and requires at least 6 weeks of data for prepaid customers. The customer segmentation component considers monthly historical data for both prepaid and postpaid customers and requires at least 6 months of data.

All the ABTs that are populated for customer retention contain monthly or weekly aggregations depending on the payment method. However, for customer segmentation, the ABTs contain monthly aggregations irrespective of the payment method.

Time Grain

ABTs are created for a business group. The business group has variables that categorize the population by payment methods (prepaid and postpaid). The time grain and the variable definitions depend on whether the model is to be built for prepaid or postpaid customers. Separate models are to be built for prepaid and postpaid payment modes. The inherently different nature of business in prepaid and postpaid requires different processing time grains. Therefore, for postpaid payment mode, monthly aggregation of variables is to be used. For prepaid payment mode, weekly aggregation of variables is to

be used. All the churn score predictions for postpaid customers would be at monthly frequency; for prepaid customers, the predictions would be at weekly grain. For customer segmentation, the data that is considered for modeling and scoring would be at monthly level for both the payment methods.

Time Window Definitions

This section focuses on the time windows that are considered while forming a churn definition. Time window lengths are crucial while defining the churn event. The lengths of time windows can affect the performance of a churn event and, as a result, the underlying model. SAS Offer Optimization for Communications enables users to choose a desired length for different time windows in the customer retention parameter configuration step. The following section explains the different time windows that would be considered for the customer retention component. Also, since business works differently in the prepaid and postpaid scenarios, these lengths would differ accordingly.

Historical data window
: the period for which data is collected and aggregated. Data that is used for prediction (subscriber behavior, payments, and so on) has to be for this period.

Churn window
: the period in which a churn event is predicted. Typically, a subscription is deactivated during this period.

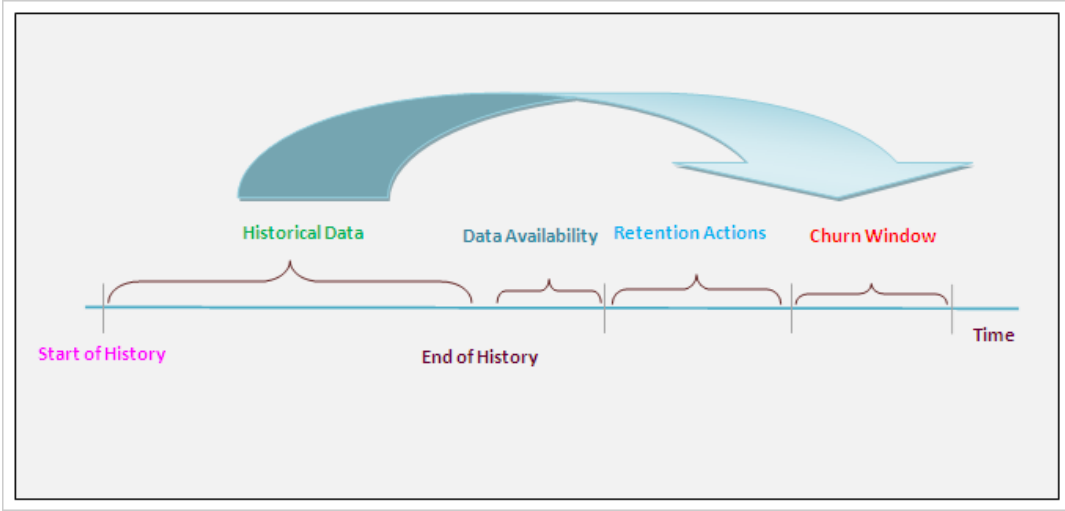
Retention period
: a time period in which a retention campaign (based on the predicted results) is organized so that a possible churn in the churn window can be avoided.

Prediction date
: the first day after the historical data period.

Data availability date
: a date when data for scoring is physically available. It is the time that is required from the end of a historical data period to get the data warehouse and ABT refreshed with data from the entire historical data period.

Gap period
: the time between the last date of the historical data and the start date of the churn window (includes retention period and data availability lag). It is the time period that the communications service provider needs to take appropriate action before the churn occurs.

Figure 5.1 *Churn Definition With Retention Window*



Defining the Target Variable

The target variable selection is the first step of the model building mode. This step is applicable only for customer retention. In this step, you have to define the target variable. The target variable depends on the payment modes (prepaid and postpaid). The functionality of selecting the target variable is provided separately for different payment modes. The definition of a business group determines the payment mode. This enables you to define the target variable. Customer retention currently targets voluntary churn. For postpaid, this phenomenon is obvious as the customer requests for disconnecting the subscription. Therefore, a fixed churn definition is provided for postpaid customers. Churn phenomena are not as obvious in prepaid as it in postpaid. There could be different criteria that mark a churn event. Customer retention enables you to select multiple criteria for defining churn. You can select different permutations (logical) of these variables to mark a churn event.

Listed below are the variables that define the target in prepaid and postpaid payment modes.

The following table gives the condition, which if true can indicate that the postpaid subscription under consideration has churned. For example, the following condition is a valid churn definition.

```
IF SUBS_STATUS_CD = DEACTIVATED or SUBS_STATUS_CD = CLOSED or SUBS_STATUS_CD = SUS
ENDED THEN

CHURN_FLG = 1

ELSE

CHURN_FLG = 0
```

Table 5.1 *Variables for Postpaid Payment Mode*

Variable Name	Description	Operator	Unit	Value
SUBS_STATUS_CD	Subscription Status Code. Active, suspended, outbound barred, and so on. This variable is also useful in deciding other target variables such as churn.	=		• DEACTIVATED • CLOSED • SUSPENDED

The following table gives various conditions, which if true can indicate that the prepaid subscription under consideration has churned. Any of the combinations with AND and OR of the conditions can be used to define the churn flag. For example, the following condition is a valid churn definition.

```
IF NUM_WKS_WO_DECR > 2 or NUM_WKS_WO_RCHRG > 3 THEN

CHURN_FLG = 1

ELSE

CHURN_FLG = 0
```

Table 5.2 *Variables for Prepaid Payment Mode*

Variable Name	Description	Operator	Unit	Value
DAYS_SINCE_LAST_RCHRG	Days since last recharge	>	week	2
WKS_WO_ANY_USAGE	Number of weeks without any usage in last 12 weeks	=	week	2
WKS_WO_IB_USAGE	Number of weeks without any incoming usage in last 12 weeks	>	week	1
WKS_WO_OB_USAGE	Number of weeks without any outgoing usage in last 12 weeks	>	week	2
NUM_WKS_WO_ddd_xxx_USAGE	Number of weeks without any ddd usage for xxx category. (Here, ddd= incoming or outgoing and downlink or uplink, xxx = LOCAL, STD, VAS, ILD, DATA, SMS, and so on.)	>	week	1

Variable Name	Description	Operator	Unit	Value
NUM_WKS_WO_DECR	Number of weeks without decrement in last 12 weeks	>	week	2
NUM_WKS_WO_RCHRG	Number of weeks without recharge in last 12 weeks	>	week	3
DAYS_TO_FIRST_ANY_USAGE_FROM_ACTVN	Number of days taken to use after subscription activation	<	day	5
DAYS_TO_FIRST_IB_USAGE_FROM_ACTVN	Number of days taken to first incoming use after subscription activation	>	day	2
DAYS_TO_FIRST_OB_USAGE_FROM_ACTVN	Number of days taken for first outgoing use after subscription activation.	>	day	2
DAYS_TO_FIRST_ddd_xxx_USAGE_FROM_ACTVN	Number of days taken for first ddd for xxx use after subscription activation. Here, ddd= incoming or outgoing and downlink or uplink, xxx = LOCAL, STD, VAS, ILD, DATA, SMS, and so on.	>	day	2
DAYS_TO_FIRST_DECR_FROM_ACTVN	Number of days taken to first decrement after subscription activation	>	day	3
DAYS_TO_FIRST_RCHRG_FROM_ACTVN	Number of days taken to first recharge after subscription activation	>	day	15
DAYS_TO_ANY_USAGE_FROM_LAST_RCHRG	Number of days taken to generate any type of usage from last recharge	>	day	4

Variable Name	Description	Operator	Unit	Value
DAYS_TO_IB_USAGE_FROM_LAST_RCHRG	Number of days taken for incoming usage from the last recharge	<	day	2
DAYS_TO_OB_USAGE_FROM_LAST_RCHRG	Number of days taken for outgoing usage from the last recharge	>	day	4
DAYS_TO_ddd_xxx_USAGE_FROM_LAST_RCHRG	Number of days taken for ddd xxx usage from the last recharge. Here, ddd= incoming or outgoing and downlink or uplink, xxx = LOCAL, STD, VAS, ILD, DATA, SMS, and so on.	>	day	2
DAYS_TO_DECR_FROM_LAST_RCHRG	Number of days taken to decrement from the last recharge	>	day	2
DAYS_SINCE_LAST_ANY_USAGE	Number of days since last usage	<	day	2
DAYS_SINCE_LAST_IB_USAGE	Number of days since last incoming usage	<	day	3
DAYS_SINCE_LAST_OB_USAGE	Number of days since last outgoing usage	<	day	2
DAYS_SINCE_LAST_ddd_xxx_USAGE	Number of days since last ddd for xxx usage (where ddd= incoming or outgoing and downlink or uplink, xxx = LOCAL, STD, VAS, ILD, DATA, SMS, and so on).	<	day	3
DAYS_SINCE_LAST_DECR	Duration in days since the last decrement. Computed as the difference in days between the last decrement from bucket date and present date.	<	day	3

Variable Name	Description	Operator	Unit	Value
VLDT_EXED_FLG	Indicator of validity period exceeded over history. It has value 1 if validity period is exceeded over history and 0 otherwise.	=		1
ZERO_BAL_FLG	Indicator of zero balance reached over history. It has value 1 if zero balance reached over history and 0 otherwise.	=		1
DECR_ MAIN_ACCT_AMT _BASE_WK_wk	Decrement amount from main account during week wk, where wk =1,2,…,12.	<	USD	10
DECR_DEDCT_AC CT_n_AMT_BASE_ WK_wk	Decrement amount from dedicated account n account during week wk, where wk =1,2,…,12 and n = 1, 2, 3…… 15.	<	USD	5

About the Scoring Mode

Overview

The scoring mode does not involve user interaction. The scoring mode assumes that the corresponding modeling ABT is created, models are built, and score codes are registered. A scoring mode is also called a batch run.

Score Code

Modeling ABT considers data that spans a historical period (in months or weeks). The modeling ABTs need not always contain recent data. In the scoring mode, scores or segment codes are generated for recent data. In model building mode, the model learns through behavioral patterns in the data and predicts the target (either churn score or segment codes) to the best possible extent. These results are stored in the form of a score code.

Score Code Application

The scoring ABT that is generated in the score code step is the input for the score code application. The score code for every model would be stored in the project workspace.

Aggregation Rules

This step ensures that the churn scores are generated at the required level. The usage revenue at subscription level is used for assigning scores to customer level..

For example, if the subscriptions are generating revenue worth 15, 85, and 25 USD per month respectively, then the churn scores can be derived by using the weighted average method.

The formula to generate the churn score using the weighted average method is as follows:

CUST_MODEL_SCORE_NUM = SUM(WT_CHURN_SCORE)/ SUM(REVENUE_AMOUNT)

Weighted churn score = Subscription probability * REVENUE_AMOUNT

Therefore, for the above example the churn scored will be calculated as follows:

CUST_MODEL_SCORE_NUM = (15*0.35+0.8*85+0.45*25)/125 = 0.676 ~ 0.68

This flags the customer as a churner, and the customer is selected in the subsequent SAS Offer Optimization for Communications workflows.

Scores Writeback

This is the last step in the scoring mode. The scores and segments for all customers are written back to the Foundation data mart by running a macro. This macro writes the scores and model information in the Foundation mart tables such as ANALYTICAL_SGMT_DTL (for customer segmentation) and CUST_MODEL_SCORE_DTL and SUBSCRP_MODEL_SCORE_DTL (for customer retention). Also, information is written in the PREDICTIVE_MODEL_RUN_DTL table (for customer retention) and SEGMENTATION_MODEL_RUN_DTL table (for customer segmentation) of the Foundation data mart.

Chapter 6

Working With Customer Retention and Customer Segmentation

Configuring the Setup for Customer Retention and Customer Segmentation

Overview

Before you build the analytical models for customer retention and customer segmentation, you have to complete the following tasks.

Initialize Parameters

The analytical models for customer retention and customer segmentation require a set of parameters. In order to configure these parameters, you have to run the cfd_analytics_param_declaration_job job. The cfd_analytics_param_declaration_job initializes the parameters that are required for customer retention and customer segmentation models. This job refers to the CFDN_ANALYTICS_CONFIG_PARAM table that resides in the CFDCONF library. The CFDN_ANALYTICS_CONFIG_PARAM table contains names and values of the global parameters. For details, see "Parameters for Customer Analytics" on page 252.

After you run the cfd_analytics_param_declaration_job job, the cfdn_initialization_crcs.sas parameter file is generated.

To initialize the parameters, complete these steps:

1. Verify the values of all the parameters in the CFDN_ANALYTICS_CONFIG_PARAM table and modify the values if necessary.
2. Log on to SAS Data Integration Studio and connect to a profile.
3. On the **Folders** tab, expand **Products** ⇨ **SAS Offer Optimization for Communications** ⇨ **Jobs** ⇨ **Analytics Setup Job Group**.
4. Run the cfd_analytics_param_declaration_job job.
5. Close SAS Data Integration Studio.
6. Go to the **`<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode`** folder.
7. Verify that the cfdn_initialization_crcs.sas initialization file is created at the specified location.

Populate ABT Data for Sample Model

The sample model that is packaged will use the prepackaged ABT data. You have to update and run INSERT scripts for ABT data population.

To update and run INSERT scripts, complete these steps:

1. Create **`samplecs`** and **`samplecr`** folders in the following folders for customer segmentation and customer retention respectively:
   - **For customer segmentation: `<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs`**

- **For customer retention:** **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr**

.

2. Create the **abt** folder in the following folders:
   - **For customer segmentation:** **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/samplecs**
   - **For customer retention:** **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/samplecr**
3. Go to the **<SAS configuration path>/Lev1/<SAS Application Server context name>** folder.
4. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
5. Declare the ABT library in Base SAS to point to the following locations depending on whether you are building the model for customer retention or customer segmentation:
   - **For customer retention:****<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/samplecr/abt**
   - **For customer segmentation:** **<SAS configuration ath>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/samplecs/abt**

   Note: Perform steps from 6 to 11 for customer retention.
6. Open the ca_insert_for_abt_crpostsub.sas macro. This macro is located in the following folder: **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts**
7. Verify the macro arguments.

   ```
   %ca_insert_for_abt_crpostsub(cfdn_crlib_nm=abt, cfdn_crabt_nm= bpp_crpost_cust);
   ```
8. Enter appropriate values for the macro arguments.
   a. Specify the abt value for the cfdn_crlib_nm argument.
   b. Specify the bpp_crpost_cust value for the cfdn_crabt argument.
9. Click **Save**.
10. Click **Run**.
11. To ensure successful execution of the INSERT scripts, view the log.

    Note: Perform steps from 12 to 17 for customer segmentation.
12. Open the ca_insert_for_abt_cspostcust.sas macro. This macro is located in the**<SAS installation path>/SASFoundation/9.2/comfdnsrv/sasmisc/Controlscripts/insertscripts** folder.

13. Verify the macro arguments.

```
%ca_insert_for_abt_cspostcust(cfdn_crlib_nm=abt, cfdn_csabt_nm= bpp_cspost_cust
);
```

14. Enter appropriate values for the macro arguments.
    a. Specify the abt value for the cfdn_cslib_nm parameter.
    b. Specify the bpp_cspost_cust value for the cfdn_csabt_nm parameter.
15. Click **Save**.
16. Click **Run**.
17. To ensure successful execution of the INSERT scripts, on the menu select **View** ⇨ **Log**.
18. Close Base SAS.

Import Sample Models

Samples models for customer retention and customer segmentation are prepackaged with SAS Offer Optimization for Communications. The purpose of these models is to demonstrate their analytical flow in SAS Enterprise Miner. Also, the extension node that is packaged for these models populates the tables that are defined for DIM and DABT_ADM libraries.

Note: Make sure that you do not run the extension node (also called writeback node) of the sample model.

The sample model for customer churn is packaged as sample_cr_post_sub.spk; the sample model for customer segmentation is packaged as sample_cs_post_cust.spk.

To install and configure the sample models:

Install Sample Models

1. Start the SAS Analytics Platform server.
2. Log on to SAS Enterprise Miner and connect to the appropriate server.
3. On the menu, select **File** ⇨ **New** ⇨ **Project**. The Create New Project dialog box appears.
4. Select the appropriate server and click **Next**.
5. In the **Project Name** box, type *cs_model* or *cr_model* depending on whether you are importing the customer segmentation or customer retention model and click **Next**.
6. In the **SAS Server Directory** box, type the appropriate path for the customer retention or the customer segmentation model as *<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/samplecr/model* or *<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/samplecs/model* respectively and click **OK**.
7. Do not change the default values for the remaining steps and click **Next** until you reach the last screen.
8. Click **Finish**. In the SAS Enterprise Miner window, the new project appears at the top left of the **Project** panel.

Configure Sample Models

9. Select the project and then select the project start-up code. The Project Start Code window appears.
10. To use the sample model with the sample data sets, define the library that contains the sample data sets.
    a. Select the Project Start Code window.
    b. Type the following code for customer retention:

    ```
    libname abt "<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2
    /Data/common_data/analytics/cr/samplecr/abt";
    ```

    Type the following code for customer segmentation:

    ```
    libname abt "<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2
    /Data/common_data/analytics/cs/samplecs/abt";
    ```

    c. Click **Run Now**.
    d. Click **OK**.
11. Right-click **Data Source** and in the Property window, select **Create Data Source**.
12. In the Data Source Wizard, complete these steps:
    a. Select the default value (SAS Table) for the metadata source and click **Next**.
    b. Click **Browse** to select a SAS table. All the predefined SAS libraries are displayed.
    c. Select the ABT library depending on whether the purpose of the ABT is customer segmentation or customer retention.
        - **For customer segmentation:** bpp_cspost_cust
        - **For customer retention:** bpp_crpost_cust
    d. Click **OK**.
    e. Click **Next**. The table properties window appears.
    f. Click **Next**.
    g. Select **`Advanced`** from the metadata advisor options and click **Next**. The details of variables included in the ABT are displayed.
    h. Set the role of the variables as explained below:

    Note: Perform this step only if you are building the ABT for customer retention. The following table indicates the roles of only the important variables. For the other variables, the role can be ID or Input.

    Table 6.1 *Roles for CR ABT Variables*

Variable name	Role
CHURN_FLG	Target
TOT_REV_AMT	Input

    i. In the Decision Configuration page, select **No**, and then click **Next**.
    j. Click **Next**. The information about the ABT variables is displayed.
    k. Click **Finish**.

13. On the menu, select **File** ⇨ **Open Model Package**.
14. Click **Browse** and locate the sample_cr_post_sub.spk file. On a Windows machine, this file is available in the **`cisbpp/misc/sampleeminermodels/segmentationmodel`** path. However, for a UNIX machine, if the spk file is on another machine, SAS Enterprise Miner must have network access to the file, or you must copy the file to your machine.
15. Select the sample_cr_post_sub.spk.spk file, and then click **Open**. The model diagram is displayed.
16. Right-click the diagram and on the **Actions** menu, select **Recreate Diagram**.
17. In the Input dialog box, type a name for the diagram, and then click **OK**.
18. Click **Yes** in the Confirmation dialog box. The model diagram is displayed.
19. Right-click the **Diagrams** option in the project tree and select **Save as** from the menu. Select a suitable location to save the diagram.
20. Type the filename as *segm_diag*. Make sure that the file type is **`xml`**. This diagram will be used when you create a customer segmentation model. For details, see "Create a Segmentation Model in SAS Enterprise Miner" on page 107.
21. Repeat steps 19 and 20 for the customer retention sample model. Type the filename for the diagram as *churn_diag*. This diagram will be used when you create a customer retention model. For details, see "Create a Customer Retention Model in SAS Enterprise Miner" on page 104.

Creating ABTs for Customer Retention and Customer Segmentation

Prerequisites

Before you create the ABTs for customer retention and customer segmentation, make sure that the following tasks are complete:

1. Populate the configuration area of the Analytics data mart with correct data. For details, see "Populating the Configuration Area of Analytics Data Mart" on page 42.
2. Familiarize yourself with the worksheets and the columns in the ABT-specific workbook. For details, see "About the ABT-Specific Workbook" on page 61.
3. Familiarize yourself with the recent behavioral, supplementary, and derived variables by reading through the Analytics data model. For details, see *SAS Offer Optimization for Communications: Analytics Data Mart*.
4. Refer to the code of the cfdn_outer_macro.sas macro. This macro is located in the **`<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts`** folder. This macro contains the sample calls for all the macros, which need to be run in order to create the ABTs. When you perform the instructions that are explained below, make sure that you have opened the cfdn_outer_macro.sas macro. Also, the instructions specify the sequence in which these macros are to be run. Before you run the macros from this macro, make sure that you verify the argument values that these macros require.

Acquire the Purpose-Specific ABT Configuration File

Depending on the purpose for which you are defining the ABT template, you have to use the correct ABT configuration workbook.

Table 6.2 *Purpose-Specific ABT Configuration Workbooks*

Purpose	Purpose Short Name	Workbook Name
Customer segmentation postpaid	CSPOST	ABT_SPECIFIC_CS_postpaid.xls
Customer segmentation prepaid	CSPRE	ABT_SPECIFIC_CS_prepaid.xls
Customer retention postpaid	CRPOST	ABT_SPECIFIC_CR_postpaid.xls
Customer retention prepaid	CRPRE	ABT_SPECIFIC_CR_prepaid.xls

Make sure that you have downloaded these workbooks and saved them in an appropriate location on your computer. For details, see "Download the Analytics Workbooks" on page 41.

Note: This section refers to the ABT_SPECIFIC_CS_postpaid.xls workbook wherever necessary. However, you have to refer to the appropriate workbook according to the purpose for which you are defining the ABT template.

Create User in SAS Management Console

The user who executes the INSERT scripts that are mentioned in the instructions below needs access to the libraries that are predefined in the SAS environment. In order to enable the user to have this capability, make sure that a user is defined in SAS Management Console with the following role and group:

- Role: Metadata Server: Unrestricted
- Group: Communication Common Oracle User Group

Update the Name and Location of the Configuration File

After you have acquired one of the above appropriate workbooks, complete the following steps:

1. Go to the **<SAS configuration path>/Lev1/<SAS Application Server context name>** folder.
2. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
3. Open the cfdn_outer_macro.sas file, which is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts** folder.

4. Add the following code:

   ```
   %let abtxls =<path in which you have stored the purpose-sepcific
   workbook>
   ```

   For example, you can add the following lines of code if you are working with the ABT_SPECIFIC_CS_prepaid.xls workbook:

   ```
   %let abtxls =C:/BPP/common/docs/docs_2/final/ABT_SPECIFIC_CS_postpaid.xls
   ```

5. Click **Save**.
6. Run the line of code that you have added in step 4.
7. To ensure successful execution of the code, on the menu select **View** ⇨ **Log**.

Define the Population

When you create an analytical model, you have to define the population for which you are building the analytical model. The subset criteria feature of DABT enables you to define your population. The subset conditions might differ depending on the business group definitions.

Note: One selection criterion is associated with only one modeling ABT and, therefore, only one DABT project.

To define the population for your analytical model:

1. Open the ABT_SPECIFIC_CS_postpaid.xls file.
2. In the **subset_where** worksheet, populate appropriate values for the parameters. For example, the WHERE_CLAUSE_EXPRESSION parameter determines the period for which historical data is required for building the ABTs.
3. Save and close the file.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_SUBSETQUERY_IMPORT
   - %CFDN_CREATE_SUBSET_WRAPPER.

   As a result, subset queries are entered into the SUBSET_QUERY_MASTER table of Analytics data mart.

Create a DABT Project

You have to define a DABT project in order to build a solution-specific ABT. A DABT project caters to only one modeling ABT.

To create a DABT project:

1. Open the appropriate workbook (for example, the ABT_SPECIFIC_CS_postpaid.xls file).
2. In the **Project** worksheet, specify the project details such as PROJECT_NM, LEVEL_NM, DESCRIPTION, LOCATION_PATH, Query_Nm, and Purpose.
3. Save and close the file.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_PROJECT_IMPORT

- %CFDN_NEW_PROJECT_WRAPPER

The PROJECT_MASTER table of the Analytics data mart is populated with the details of the new project.

Create a Project Workspace

Each project has a workspace in which the modeling and scoring ABTs, configuration details, and score code are stored.

To create a workspace for a project, run the %CFDN_CREATE_PROJECT_WORKSPACE macro. This macro has the argument cfdn_crcs_data_path. Specify the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics** value for this argument. A folder is created in this path. The name of the folder that is created is the same as the name of the DABT project. This will be referred to as **<Project_nm>** in the subsequent paths when you create the project. In addition, subfolders **abt**, **art**, **model**, and **srcd** are created in this folder.

This macro is based on the following assumptions:

- This macro reads values of ABT_NM, PROJECT_NM, and PURPOSE from the d_project temporary table of the scrflx library and creates the following folders in the project workspace:
  - abt
  - art
  - scrd
  - model
- The ABT resides in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/<Project_nm>/abt** folder or the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/<Project_nm>/abt** folder, depending on the solution and purpose for which the ABT is built. All modeling and scoring ABTs reside in the same location.
- The SAS Enterprise Miner project name is user-specific. The DABT project is different from the SAS Enterprise Miner project.
- All SAS Enterprise Miner projects that are built on a given modeling ABT reside in the DABT project workspace of a modeling ABT (that is, in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/<Project_nm>/model** folder).

  For example, assume that the **data** folder resides in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics** folder. If the ABT_NM is SEGMENTATION_DATA, Purpose is CSPOST, and PROJECT_NM is Segmentation Project, then the folder structure is **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/segmentationproject/model**.

Therefore, a user can build a SAS Enterprise Miner project with this ABT with the name, *Segmentation for postpaid customers*. The SAS Enterprise Miner project workspace is created in the following folder:

<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/Segmentation Project/model/segmentation for postpaid customers

- The project workspace creation macro calls two integration macros. These macros are executed for all customer retention and customer segmentation projects.

  %cfdn_new_project_specific_lib
  After you create the project workspace, this macro generates the library name and libref. It also, makes an entry of the ABT library for that project in the LIBRARY_MASTER table of the Analytics data mart. In addition, this macro creates an entry against the LOCATION_PATH for that project in the PROJECT_MASTER table of the Analytics data mart.

  %cfdn_create_include_libsfile
  This macro creates the libs_include.sas file in the **scrcd** folder of the project workspace. This file contains a declaration of the ABT library that is generated when you run the %cfdn_new_project_specific_lib macro.

Update the Library Reference

In this step, the user has to manually input the library short name that gets created in the "Create Project Workspace" step to the modeling ABT to be created. The user has to make this entry in the ABT_SPECIFIC_CS_postpaid.xls file in the **ABT** worksheet area under the library field.

To update the library reference:

1. Open the ABT_SPECIFIC_CS_postpaid.xls file.
2. In the **ABT** worksheet, for the library column, enter the library short name that is created when you create a project workspace.
3. Save and close the file.

Define Modeling ABT

This step makes the following assumptions:

- The library reference and the LIBNAME statement that are entered in the **ABT** worksheet should be the same as those entered in the **Project** worksheet of the ABT_SPECIFIC_CS_postpaid.xls file.
- The libref should point to a location in which the modeling ABT would reside after you create it. This is the ABT folder under a DABT project workspace. That is, it should point to the following locations:
  - **For customer retention:** **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/<Project_nm>/abt**

- **For customer segmentation:** **`<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/<Project_nm>/abt`**

To define a modeling ABT:

1. Open the ABT_SPECIFIC_CS_postpaid.xls file.
2. In the **ABT** worksheet, provide details such as Table_Nm, Short_Nm, Description, Target_Time_Freq, Target_Time_Freq_Count, Library, LIBRARY_REFERENCE, LIBNAME_STATEMENT, Level, PROJECT_NM, FLG, Purpose, and ABT_TYPE.
3. Save and close the file.
4. In Base SAS, run the following macros from the %cfdn_outer_macros.sas macro.
   - %CFDN_ABT_IMPORT
   - %CFDN_NEW_ABT_WRAPPER

   As a result, a new record of the modeling ABT that is to be created with respect to the DABT project is added in the ABT_MASTER table.

Define ABT Variables

To define variables for the modeling ABT:

1. Open the ABT_SPECIFIC_CS_postpaid.xls workbook.
2. In the **Supplementary**, **Recent**, and **beh** worksheets, specify appropriate values for the columns. For details, see "About the ABT-Specific Workbook" on page 61.
3. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro.
   a. Run the %cfdn_abtvars_import macro. This macro converts the data from the worksheet to a SAS data set.
   b. Run the %cfdn_abt_dimval_insert macro. This macro creates temporary SAS data sets for each type of variable that is mentioned in the macro.

      Note: Make sure that you run this macro only for Behavioral (BEH) and Recent (RNT) variables.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_CREATE_BEH_VAR_WRAPPER
   - %CFDN_CREATE_RNT_VAR_WRAPPER
   - %CFDN_CREATE_SPM_VAR_WRAPPER

   This step creates data sets for the supplementary, recent, and behavioral variables in the **`<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/dabt_data/scratch/flex`** folder. These data sets are created in the scrflx library (for example, the beh_var_list_15.sas7bdat and beh_var_list_15_edit.sas7bdat).
5. In all the *_edit.sas7bdat data sets, verify the variable names. If the length of the variable names exceeds the maximum limit, modify the variable names.

   Note: The value for edit_cd and edit_option columns indicate whether the length of the variable names exceeds the maximum limit of 28 characters. Rename all the variables where edit_option is set to D (representing delete as the length is greater than 28 characters). Edit names of all such variables and change the

edit_option for those to A to represent “accept” because the length is less than or equal to 28 characters.

6. To save the variables, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_SAVE_BEH_VAR_WRAPPER
   - %CFDN_SAVE_RNT_VAR_WRAPPER
   - %CFDN_SAVE_SPM_VAR_WRAPPER

   This step reads data from the temporary tables and enters data in the following tables:
   - VARIABLE_MASTER
   - SUPPLEMENTARY_VAIABLE
   - RECENT_VARIABLE
   - BEHAVIORAL_VARIABLE
   - ABT_X_VARIABLE
   - VARIABLE_DIM_ATTRIBUTE_FILTER

Define Derived Variables

Before you create derived variables, make sure that you have complete information about them. For details, see “Derived Variables” on page 259.

To create derived variables:

1. Open the ABT_SPECIFIC_CS_postpaid.xls workbook.
2. In the **derivedvardetails** and **derivedvarsource** worksheets, specify appropriate parameter values.
3. Save and close the file.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_DRVDVARS_IMPORT
   - %CFDN_SAVE_DRVD_VAR_WRAPPER

Define Target Variable

A target variable is defined only for modeling ABTs. DABT enables you to define the target variable according to your requirements. A target variable is a derived variable. You can define the target variable through the **derivedvardetails** and **derivedvarsource** worksheets of the corresponding ABT_SPECIFIC.xls file. You have to define a target variable only for predictive models such as customer retention models. You need not perform this step if you are defining a customer segmentation model.

When you build a modeling ABT for customer retention, customers are observed for deactivations. These deactivations are observed for a certain period from the end of the historical period. This period is called the target period. A TARGET_PERIOD_CNT is associated with the ABT at the time of definition. This period indicates the duration of the target period. The frequency of the target period can be in months, weeks, or days. A

target period frequency is also associated with the ABT at the time of definition. For details, see the **ABT** worksheet in "About the ABT-Specific Workbook" on page 61.

To define a target variable:

1. Open the ABT_SPECIFIC_CS_postpaid.xls sheet.
2. In the **derivedvardetails** and **derivedvarsource** worksheets, specify appropriate parameter values.
3. Save and close the file.
4. Find the sk of the ABT that you are building and the sk of the target variable in that ABT. You can do so by searching for the physical name of the variable and the ABT in the VARIABLE_MASTER and ABT_MASTER tables.
5. In Base SAS, run the %outcome_var macro from the %cfdn_outer_macro.sas macro. This macro sets the value of the PART_OF_ABT_FLG column in the ABT_X_VARIABLE table to Y. For example, a sample call to this macro can be as follows:

   ```
   %outcome_var(m_abt_sk=1, m_abt_var_sk=108);
   ```

   Note: If a variable is chosen as outcome variable, then it must be a part of that ABT. In other words, the part_of_abt_flg must be set to Y for that variable. This indicates that unless a variable is marked to be computed for an ABT, it cannot be marked as an outcome variable.

Select ABT Variables

To select the variables that are to be included in customer retention and customer segmentation ABTs, the editable data set corresponding to the ABT_X_VARIABLE table and to the ABT under consideration is first created. The ABT_X_VARIABLE_edit_* data set is created in the scrflx library. The part_of_abt_flg in this data set can be set to Y or N depending on whether the corresponding variable is to be included in the ABT.

After the data set is edited, the changes that are made have to be written back to the ABT_X_VARIABLE data set.

To complete the above mentioned tasks:

1. Identify the name and the SK of the ABT that you are working with.
2. Identify variables that are to be excluded or included in the ABT and make a list of their SKs.
3. In Base SAS, run the % cfdn_create_edit_abt_x_var macro from the %cfdn_outer_macro.sas macro. This macro creates the ABT_X_VARIABLE_edit_* data set corresponding to the ABT_X_VARIABLE table of the scrflx library.
4. In Base SAS, open the ABT_X_VARIABLE_edit_* data set.
5. Set the value of part_of_abt_flg to Y or N according to your requirement.
6. In Base SAS, run the %CFDN_UPDATE_ABT_X_VAR_LIST macro from the %cfdn_outer_macro.sas to write these changes to DABT_ADM.ABT_X_VARIABLE table.

Construct ABT

You have to run the %build_abt macro for building the ABT. Specify the ABT_SK of the ABT that is to be constructed and the date on which the ABT is constructed in the macro call.

To construct the ABT:

1. In Base SAS, open the build_abt macro. The call for this macro is in the cfdn_outer_macro macro.
2. Specify appropriate values for the following arguments. These arguments are a part of a deployed ETL job that the build_abt macro calls.

   bpp_prj_tmp_data_dir
   : path that is specific to the SAS Offer Optimization for Communications application and is not used for customer analytics. In the autoexec file, you can set the value for this parameter as NULL. The application dynamically sets the value as required.

   bpp_prj_abt_dir
   : path that is specific to the SAS Offer Optimization for Communications application and is not used for customer analytics. In the autoexec file, you can set the value for this parameter as NULL. The application dynamically sets the value as required.

   trg_prd
   : value for the target period count for the ABT, which is to be built.

   bg_id
   : ID for the business group that is used in the subsetting criterion.

   m_abt_sk
   : surrogate key of the ABT that you want to build.

   m_abt_bld_dt
   : the date on which the ABT needs to be built. The historical data that will be used for computing variables will belong to the period that ends a day before this date.

3. Run the %build_abt macro. For example, the code can be as mentioned below.

```
option mlogic mprint symbolgen;
%let bpp_prj_tmp_data_dir =C:\SAS\Config\Lev1\Applications\SASOfferOptForCommServer5.2
\Logs\dabt_logs\flex;
%let  bpp_prj_abt_dir =C:\SAS\Config\Lev1\Applications\SASOfferOptForCommServer5.2
\Logs\dabt_logs\flex;
%let bg_id=9_BUSINESS_GROUP_ID;
%let trg_prd = 1;
%build_abt(m_abt_sk = 31, m_abt_bld_dt=20APR2009);
```

Example: Building a Modeling ABT for Customer Retention

Assume that the ABT_SK is 10, ABT name is sample_abt, the corresponding DABT project name is sample_prj, and the purpose of the ABT is customer retention. After you run the %build_abt macro, the following steps are executed:

- The high-level logs, build_abt_10 and build_subset_criteria_10 are created at the location: **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Logs/dabt_logs/flex**.

- The modeling ABT is created in the following folder: **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/sample_prj/abt** .
- The intermediate data sets are created in the following folder:**<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/dabt_data/scratch/cm/10**. This location will be accessible in the Base SAS session as a SAS library, named work_area.
- The PARAMETER_LIST_HM data set that is created at the above location contains the error codes for all the steps that are executed in order to build the ABT. The error codes are stored in the RETURN_CD column of the PARAMETER_LIST_HM data set. The value of the RETURN_CD column corresponding to each JOB_NM column should be 0 for building the ABT successfully.
- The logs corresponding to values of all JOB_NM columns for the PARAMETER_LIST_HM data set are stored in the following folder: **SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Logs/dabt_logs/schedule_cm/sample_abt**.

Extension Node for Sample SAS Enterprise Miner Models

Assumptions

The extension node for these models is based on the following assumptions:

- A model is uniquely identified by its model ID. A model ID contains the metadata ID for the SAS Enterprise Miner project through which the model is being built.
- All errors are tracked either through pop-up messages that are displayed in SAS Enterprise Miner or through the session log. Therefore, error handling is not separately performed for the extension node.
- A diagram in SAS Enterprise Miner has only one writeback node.
- A SCORE node must precede the writeback node.
- The SAS Enterprise Miner project is located in the workspace under the **model** folder of the DABT project workspace with the SAS Enterprise project name folder. For example, the customer retention project can be located at the following location: **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/RetentionProject/model/Customer Retention for postpaid customers**. Similarly, the customer segmentation project can be located at the following location: **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/SegmentationProject/model/Customer Segmentation for postpaid customers**
- The source ABT path is the library name or path against the modeling ABT. For example, depending on the purpose for which the ABT is built, the path can be as follows:

  - **For customer retention:** `SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/<project_nm>/abt`
  - **For customer segmentation:** `SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/<project_nm>/abt`
- For a modeling ABT, users can build multiple projects in SAS Enterprise Miner. However, after the models are assessed, users can register only a single model. As a result, only one model is in production and only one set of significant variables are available for a certain modeling ABT.
- The optimized score code has a consistent name for all the modeling ABTs and is located in the **scrcd** folder of the DABT project workspace. The score code is stored with the name optimizedcode.sas . The code is developed for a combination of a certain model that is to be in production and a modeling ABT. Depending on the purpose for which the model is built, this code is stored at the following location:
  - **For customer retention:** `SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/<project_nm>/scrcd`
  - **For customer segmentation:** `SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/<project_nm>/scrcd`
- The extension node should be run only when the user completes all activities in SAS Enterprise Miner.
- Running the extension node multiple times can cause the addition of multiple model-related entries in the Foundation data mart. When you want to rerun the extension node, in the SAS Enterprise Miner start-up code, set the value of cfdn_reexecute_flg to Y.
- No versions of a modeling ABT are stored. Whenever there is a need to retune the model, users can copy the ABT to a suitable location for assessment purpose.
- No versions of the models are stored in the Foundation data mart. Whenever a model is refreshed, it is considered as a new model. Therefore, the record of the old model is marked as closed and a new record is inserted for the new model.

Functionality of the Extension Node

In order to cater to the scoring mode requirements, the extension node supports the following functionality:

- When the extension node is run for the first time, it searches for the record of the model in the Foundation data mart. If the record exists, then the model is being updated. It closes the earlier record and enters a new record for the same model. If the record does not exist, then it indicates that a new model is being built. In this case, a new entry is made in the model with corresponding model details. These details can be user ID, model ID, model creation date, model type code, model name, level of model (customer, account or subscription), and model description. The CFDDIM.ANALYTICAL_MODEL_D table is updated every time the node is run.
- A macro is called from the extension node. As a result, the corresponding scoring project is created. If the name of the DABT project that is created for the modeling

ABT is **<project_nm>**, then the name of the scoring project will be **<project_nm>_scr**.

- The optimized score code of the model is written as a SAS file (with the name optimizedcode.sas) and a SAS catalog in the following location:
  - **For customer retention:** **<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/<project_nm>/scrcd**
  - **For customer segmentation:** **<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/<project_nm>/scrcd**
- The significant variables of the model are written into the ABT_SGNFCNT_VARS data set at the following location:
  - **For customer retention:** **<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/<project_nm>/scrcd**
  - **For customer segmentation:** **<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/<project_nm>/scrcd**
- The %cfdn_new_analytical_model DABT macro makes an entry of the analytical model that is built on top of a modeling ABT into the Analytics data mart. This macro writes the model information into the ANALYTICAL_MODEL_MASTER and ABT_MASTER table of the Analytics data mart. The ABT_MASTER table is updated with the latest MODEL_SK of the model that is built on that particular ABT. In the re-execution mode, the new entry of the model is made in the ANALYTICAL_MODEL_MASTER table of the Analytics data mart. Also, the latest MODEL_SK is written in the ABT_MASTER table
- The %cfdn_scr_abt_var_dts_wb_nonbpp macro is called through the node. This macro makes an entry of the scoring ABT for a given model and writes the significant variables of the model into the ABT_X_VARIABLE table of the Analytics data mart. It also makes an entry of the scoring ABT for that modeling ABT and links it to the scoring project. Remaining details of the modeling project and modeling ABT are copied for the scoring project and scoring ABT. If the model is built at subscription level and the scores are to be aggregated at customer level, then the revenue measure needs to be a part of the scoring ABT. This macro assigns the corresponding average revenue amount variable to the corresponding scoring ABT even if it is not a part of the significant variables. This assignment is done in the ABT_X_VARIABLE table of the analytics data mart. This completes the modeling part.
- A model can deteriorate over a period of time for reasons such as changes in marketing strategies, promotion of new plans, and addition of new rules and regulations. Such changes trigger retuning (also called re-execution) of the analytical model. Re-execution includes changing the data in the modeling ABT or changing the modeling strategies. When an analytical model is re-executed, the ABT_X_VARIABLE table is updated with the new significant variables of the model. Also, when a model is rebuilt or recalibrated, set the value of CFDN_REEXEC_FLG to Y.

Building a Customer Retention Model

Prerequisites

Before creating a customer retention model, make sure that a suitable user ID is defined for performing tasks in SAS Enterprise Miner. For details, see *SAS Offer Optimization for Communications: Installation and Configuration Guide*.

Also, ensure that libraries DABT_ADM, CFDDIM, CFDFACT, and CFDMISC are available in the SAS Enterprise Miner environment. If these libraries are not available, then declare them in the SAS Enterprise Miner start-up code.

In addition, make sure that a modeling ABT is created through DABT, and the DABT project workspace exists. For details, see "Create a DABT Project" on page 94.

The following sections refer to the DABT customer retention project as **`churn_project`**.

Create a Customer Retention Model in SAS Enterprise Miner

Log On to SAS Enterprise Miner

To create a churn model, log on to SAS Enterprise Miner with a profile and connect to a server.

Create a Project

1. Click **New Project**.
2. In the Create New Project wizard, complete these steps:
   a. Select the **SAS Server** and click **Next**.
   b. In the **Project Name** box, type *Churn Model for Postpaid*.
   c. In the **SAS Server Directory** box, enter the path for the **`MODEL`** folder that is created in the DABT project workspace. For example, you can type the path as **`<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/churn_project/model`**.
   d. Click **Next**.
   e. Do not change the default folder location that is displayed for **SAS Folder Location** and click **Next**.
   f. Click **Finish**. This step creates the SAS Enterprise Miner project and also the project workspace.
3. Select the *Churn Model for Postpaid* project and then in the **Property** sheet, click **Project Start Code**.
4. In the Project Start Code window, copy the code from the cfdn_cr_em_startup.sas file. This code is available in the **`<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommunication5.2/SASCode/insertscripts`** folder.

Note: Make sure that you enter proper values for all the parameters in the start-up code. Whenever the SAS Enterprise Miner model details are being written back for the first time, the value of cfdn_reexec_flg should be N. For all the subsequent runs, the value of cfdn_reexec_flg should be Y.

5. Click **Run Now**.
6. Click **Log** to view the log file and make sure that the log file does not contain any errors.
7. Click **OK**.

Create Diagram

1. In the project tree, right-click **Diagrams** and select **Import Diagram from xml**.
2. Select the `churn_diag.xml` that you created when you imported the sample model.
3. Click **Open**. The diagram is imported and opened in the Diagram pane.

Create Data Source

1. Right-click **Data Source** and in the Property window, select **Create Data Source**.
2. In the Data Source Wizard, complete these steps:
   a. Select the default value (SAS Table) for the metadata source and click **Next**.
   b. Click **Browse** to select a SAS table. All the predefined SAS libraries are displayed.
   c. Double-click the library in which you stored the modeling ABT for customer retention and select the appropriate data set. For example, the data set can be CHURN_DATA. For details, see "Construct ABT" on page 100. The subsequent steps refer to this data set.
   d. Click **OK**.
   e. Click **Next**. The table properties window appears.
   f. Click **Next**.
   g. Select `Advanced` from the metadata advisor options and click **Next**. The details of variables included in the ABT are displayed.
   h. Set the role of the variables as explained below:

      Note: The following table indicates the roles of only the important variables. For the other variables, the role can be ID or Input. Make sure that the level of the target variable is binary.

      Table 6.3 *Roles for ABT Variables*

Variable name	Role
CHURN_FLG	Target
TOT_REV_AMT	Input

      Note: Churn score is computed at subscription level. However, it needs to be aggregated at customer level. The usage revenue amount at subscription level is represented in the AVG_REV_AMT column. There can be a scenario in

which a customer has multiple subscriptions. In this case, the proportion of AVG_REV_AMT at subscription level is considered to provide weights for churn scores, which need to be aggregated at customer level. Therefore, the AVG_REV_AMT column is mandatory. The column name might be different in the Analytics data mart. You need to provide the correct column name that indicates the average revenue for every subscription. This column should be populated as a part of the ABT building activity. Also, you can configure the name of this column. In the start-up code of the SAS Enterprise Miner project for which you have built the model, you have to provide the name of this column. This column name has to be same as the value of the cfdn_avg_rev_var_nm macro variable .

i. In the Decision Configuration page, select **No**, and then click **Next**.

j. Click **Next**. The information about the ABT variables is displayed.

k. Click **Finish**.

Build Customer Retention Model

1. Delete the input data set of the diagram that was created in the Create Diagram step.
2. Drag and drop the CHURN_DATA data set in the **Diagram** pane and connect it to the first node of the diagram.
3. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Train`**.
4. Delete the input data set of the diagram that is connected to the score node.
5. Drag and drop the CHURN_DATA data set in the **Diagram** pane and connect it to the score node of the diagram.
6. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Score`**.
7. Right-click the CHURN_DATA data set, which is connected to score node and click **Edit Variables**.
8. Set the role of the CHURN_FLG variable to **`Rejected`**.
9. Right-click the **Model Writeback** node and select **Update**.
10. After the diagram is updated successfully, right-click the **Model Writeback** node and select **Run**. Select **Yes** when asked for confirmation.
11. Click **OK** when the run is complete. In this step, the following tasks are completed:
    - The ANALYTICAL_MODEL_D table of the Foundation data mart and ANALYTICAL_MODEL_MASTER table of the Analytics data mart are updated.
    - The CHURN_DATA.sas score code is generated in the **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/churn_project/scrcd`** folder.
    - The significant variables of the model are written in the abt_sgnfcnt_vars data set. This data set is available in the following location: **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/churn_project/scrcd`**
    - The CHURN_PROJECT_SCR scoring project is registered for the CHURN_DATA modeling ABT.

- In the batch run, the CHURN_DATA_SCR scoring ABT is generated in the following location: **<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cr/churn_project/abt**.

Building a Customer Segmentation Model

Prerequisites

Before creating a customer segmentation model, make sure that a suitable user ID is defined for performing tasks in SAS Enterprise Miner. For details, see *SAS Offer Optimization for Communications: Installation and Configuration Guide*.

Also, ensure that libraries DABT_ADM, CFDDIM, CFDFACT, and CFDMISC are available in the SAS Enterprise Miner environment. If these libraries are not available, then declare them in the SAS Enterprise Miner start-up code.

In addition, make sure that a modeling ABT is created through DABT and the DABT project workspace exists. For details, see "Create a DABT Project" on page 94.

The following sections refer to the DABT customer segmentation project as **segmentation_project**.

Create a Segmentation Model in SAS Enterprise Miner

Log On to SAS Enterprise Miner

Log on to SAS Enterprise Miner with a profile and connect to a server.

Create a Project

1. Click **New Project**.
2. In the Create New Project wizard, complete these steps:
   a. Select the **SAS Server** and click **Next**.
   b. In the **Project Name** box, type *Segmentation Model for Postpaid*.
   c. In the **SAS Server Directory** box, enter the path for the **MODEL** folder that is created in the DABT project workspace. For example, you can type the path as **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/segmentation_project/model**.
   d. Click **Next**.
   e. Do not change the default folder location that is displayed for **SAS Folder Location** and click **Next**.
   f. Click **Finish** on the last page of the wizard. This step creates the SAS Enterprise Miner project and also the project workspace.
3. Select the *Segmentation Model for Postpaid* project and then in the **Property** sheet, click **Project Start Code**.

4. In the Project Start Code window, copy the code from the cfdn_cs_em_startup.sas file. This file is available in the **`<SAS installation path>/SASFoundation/9.2/comfdnsrv/sasmisc/Controlscripts/insertscripts`** folder.

   Note: Make sure that you enter proper values for all the parameters in the start-up code. Whenever the SAS Enterprise Miner model details are being written back for the first time, the value of cfdn_reexec_flg should be N. For all the subsequent runs, the value of cfdn_reexec_flg should be Y.

5. Click **Run Now**.
6. Click **Log** to view the log file and make sure that the log file does not contain any errors.
7. Click **OK**.

Create Diagram

1. In the project tree, right-click **Diagrams** and select **Import Diagram from xml**.
2. Select the **`segm_diag.xml`** that you created when you imported the sample model.
3. Click **Open**. The diagram is imported and opened in the Diagram pane.

Create Data Source

1. Right-click **Data Source** and in the Property window, select **Create Data Source**.
2. In the Data Source Wizard, complete these steps:
   a. Select the default value (SAS Table) for the metadata source and click **Next**.
   b. Click **Browse** to select a SAS table. All the predefined SAS libraries are displayed.
   c. Double-click the library in which you stored the modeling ABT for customer segmentation and select the appropriate data set. For details, see "Construct ABT" on page 100. For example, the data set can be SEGMENTATION_DATA. The subsequent steps refer to this data set.
   d. Click **OK**.
   e. Click **Next**. The table properties window appears.
   f. Click **Next**.
   g. Select **`Advanced`** from the metadata advisor options and click **Next**. The details of variables included in the ABT are displayed.
   h. In the Decision Configuration page, select **No**, and then click **Next**.
   i. Click **Next**. The information about the ABT variables is displayed.
   j. Click **Finish**.

Build Customer Segmentation Model

1. Delete the input data set of the diagram that was created in the Create Diagram step.
2. Drag and drop the SEGMENTATION_DATA data set in the **Diagram** pane and connect it to the first node of the diagram.
3. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Train`**.

4. Drag and drop the SEGMENTATION_DATA data set in the **Diagram** pane and connect it to the score node of the diagram.
5. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Score`**.
6. Right-click the **Model Writeback** node and select **Update**.
7. After the diagram is updated successfully, right-click the **Model Writeback** node and select **Run**. Select **Yes** when asked for confirmation.
8. Click **OK** when the run is complete. In this step, the following tasks are completed:
   - The ANALYTICAL_MODEL_D and ANALYTICAL_SGMT_DTL tables of the Foundation data mart and ANALYTICAL_MODEL_MASTER table of the Analytics data mart are updated.
   - The SEGMENTATION_DATA.sas score code is generated in the **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/segmentation_project/scrcd`** folder.
   - The significant variables of the model are written in the abt_sgnfcnt_vars data set. This data set is available in the following location: **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/segmentation_project/scrcd`**
   - The SEGMENTATION_PROJECT_SCR scoring project is registered for the SEGMENTATION_DATA modeling ABT.
   - In the batch run, the SEGMENTATION_DATA_SCR scoring ABT is generated in the following location: **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/cs/segmentation_project/abt`**.

Batch Run for Customer Retention and Customer Segmentation

Assumptions

The batch run is based on the following assumptions:

- If a different subsetting criterion needs to be applied to scoring ABT, then use the cfdn_update_project_subset_query macro. For details, see "Procedure" on page 110.
- A suffix _scr would be appended to the ABT name of the modeling ABT.
- Modeling and scoring ABTs reside in the **`abt`** folder of the project workspace.
- No versions of a scoring ABT would be maintained. This is because the only requirement is the scores that are generated for different scoring runs, which are used for checking the model performance.
- No versions of the optimized score code are maintained. Whenever a model is tuned, the older score code is overwritten by the new one.

Procedure

After the ABT for customer retention or customer segmentation is built, the analytical model is built in SAS Enterprise Miner. This model can be deployed for generating scores for the subsequent runs. In order to complete this task, you have to run the cfdn_batch_macro.sas batch macro.

This section details the steps that you must follow for the scores writeback (also called batch run) for the customer retention and customer segmentation modules.

1. Go to **<SAS configuration path>/Lev1/<SAS Application Server context name>**.
2. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
3. Open the cfdn_outer_macro.sas file, which is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts** folder.
4. Associate the subset query with the scoring project.

   Note: Complete the instructions for this step if you want to apply a different subset criterion (other than what you have defined for modeling ABT) to the scoring ABT.

   a. Find the call for the cfdn_update_project_subset_query macro in the cfdn_outer_macro. If you are unable to locate this call, add this line of code in the cfdn_outer_macro macro.

   b. Assign the name of the scoring project to the cfdn_project_nm macro argument.

   c. Assign the name of the subsetting query that you want to associate with the scoring project to the cfdn_query_nm macro argument.

   d. Run the cfdn_update_project_subset_query macro. View the logs through Base SAS.

   e. Verify that the appropriate subset query is associated with the scoring project in the DABT_ADM.PROJECT_MASTER table.
5. Execute the batch run for the project.

   a. The cfdn_batch_macro macro executes the scoring run of a DABT project. In order to run this macro, specify appropriate values for the following arguments:

      project_sk
      surrogate key of the project that is to be executed in the batch run.

      abt_sk
      surrogate key of the corresponding scoring ABT.

      cfdn_avg_rev_var_nm
      variable name (corresponding to the variable_column_nm of the DABT_ADM.VARIABLE_MASTER table) for the average revenue column, which will be used while aggregating the scores from subscription level to customer level. This aggregation is applicable to customer retention only. Therefore, for customer retention, the column name that appears here must be a part of the scoring ABT represented by the abt_sk provided. For customer segmentation, you can provide a NULL value for this argument.

execution_dt
the supplied value for this argument indicates the date as of which the ABT needs to be built. If a value is not supplied, then the ABT will be built with the current date.

b. In Base SAS, open a new program.

c. Type the following lines of code in the program:

For customer retention:

```
option mlogic mprint symbolgen;
%cfdn_batch_macro(project_sk=17,abt_sk=7,cfdn_avg_rev_var_nm=AVG_PSU_URVAT_L6M);
```

For customer segmentation:

```
option mlogic mprint symbolgen;
%cfdn_batch_macro(project_sk=17,abt_sk=7, cfdn_avg_rev_var_nm=null);
```

d. Save the file at a suitable location.

e. Go to **<SAS configuration path>/Lev1/SASApp/BatchServer** and invoke the sasbatch.bat or sasbatch.sh file.

f. Invoke the SAS code that you saved in step d.

g. View the log file that is created in the following folder: **<SAS configuration path>/Lev1/Applications/SASOfferOptForCommServer5.2/Logs/dabt_logs/flex/today's date**. The log file will be created with the name **analytics_batch_<scoring_project_sk><scoring_abt_sk>**. Here, the **<scoring_project_sk>** is the surrogate key of the project that you scheduled whereas **<scoring_abt_sk>** is the surrogate key of the corresponding scoring ABT.

Details

In the batch run, the following steps are executed:

1. Based on the scoring ABT sk and the execution date provided as the macro s to the cfdn_batch_run macro, the build_abt macro is called. As a result, the scoring ABT is generated in the **abt** folder of corresponding DABT project workspace.
2. After the scoring ABT is generated, the corresponding optimized score code is applied. There is always only one score code available for a given scoring ABT. Therefore, the optimizedcode.sas code is applied on the scoring ABT.
3. After the scored ABT is created, the scores are written back to the Foundation data mart by running the cfdn_batch_macro macro. The cfdn_batch_macro searches for the purpose of the project under which the modeling ABT is created, call the corresponding macro, and updates the Foundation data mart tables accordingly.

   If the purpose of the model is customer retention, then the churn scores are written back to the SUBSCRP_MODEL_SCORE_DTL table of the Foundation data mart. If the scores aggregation flag is set to Y, then these scores are aggregated to customer level. The aggregated scores are then written back to the CUST_MODEL_SCORE_DTLCUST table of the Foundation data mart. The information of the scoring run such as the analytical model sk and the score date is written back to the PREDICTIVE_MODEL_RUN_DTL table, every time the model is scored.

   If the purpose of the model is customer segmentation, then the segment codes are written back to the CUST_ANALYTICAL_SGMT_DTL table of the Foundation

data mart . The information about the scoring run such as the analytical model sk and the score date is written back to the SEGMENTATION_MODEL_RUN_DTL table, every time the model is scored.

Chapter 7

Introduction to Cross-Sell and Up-Sell

Overview of Cross-Sell and Up-Sell

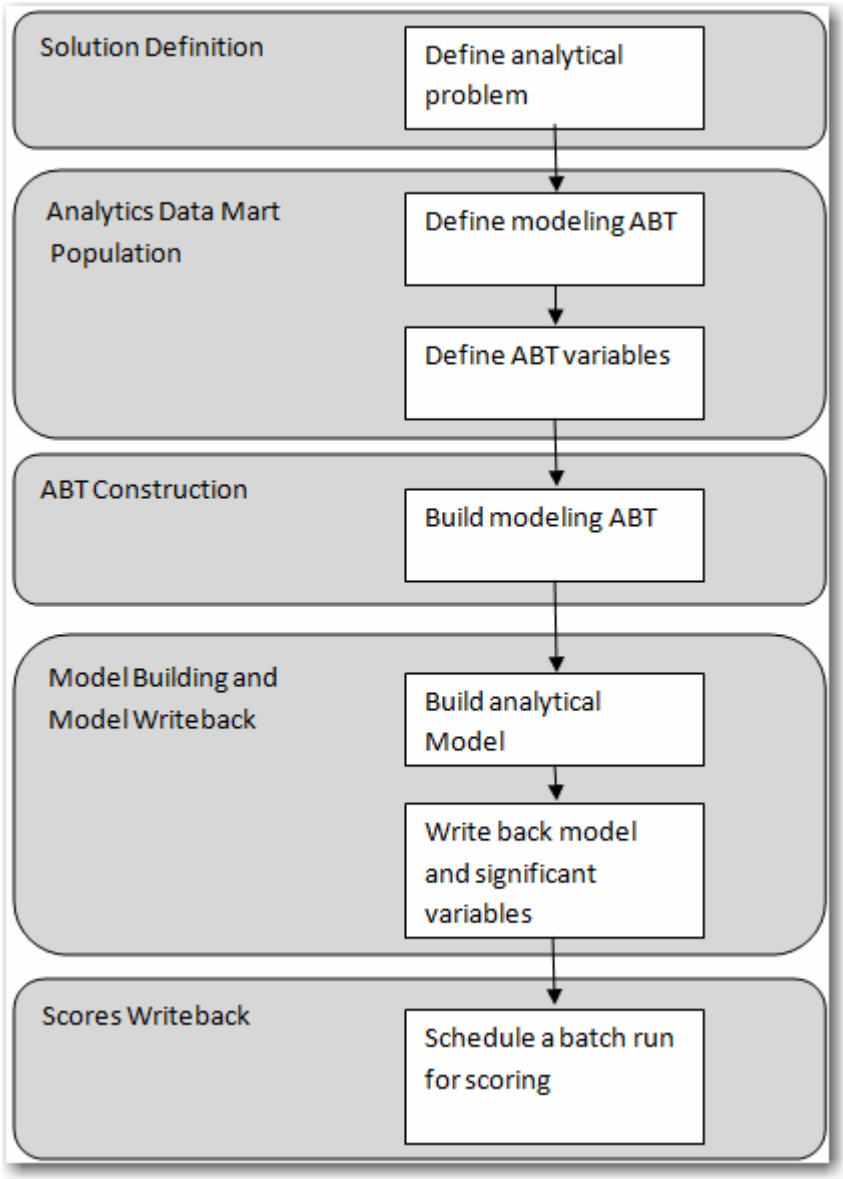
Cross-sell and up-sell is an analytical component of SAS Offer Optimization for Communications. Cross-sell is a sale of additional (supplemented) products to existing customers. Up-sell is a sale of higher value products or services to existing customers. The objective of cross-sell and up-sell is to maximize the revenue that is gained from each customer and thereby increase the value of customers. As a result, this module aims to increase customer satisfaction.

Using appropriate analytical techniques, the cross-sell and up-sell component derives the probability of a customer accepting a particular offer or service that is recommended to him or her. The cross-sell and up-sell scores are produced as an output of this module. After you select the offers and services for cross-sell and up-sell, you can build the model at customer or subscription level respectively. However, the scores are produced for a combination of customer and offer or subscription and service.

Overview of the Analytical Process Flow

The following diagram illustrates the analytical process flow for cross-sell and up-sell.

Figure 7.1 *Analytical Process Flow*



The subsequent topics give an overview of each step of the process flow.

Solution Definition

Defining the Analytical Problem

Overview

When you define the analytical solution, you have to consider the following aspects:

- the business problem for which the analytical model is to be built
- the target population on which the analytical model is to be built
- the level and time grain at which the analytical model is to be built
- the historical and scoring time window definitions

Business Problem Definition

A cross-sell and up-sell model can be built for any one of the objectives:

- To derive the cross-sell and up-sell score that indicates the probability of a customer accepting the proposed offer.
- To derive the cross-sell and up-sell score that indicates the probability of activating the proposed service for a subscription.

Population Definition

For defining the population for a cross-sell and up-sell model, you have to first consider whether you want to use cross-sell and up-sell as a stand-alone module or as an analytical component of SAS Offer Optimization for Communications.

In SAS Offer Optimization for Communications, the cross-sell and up-sell score is one of the variables based on which business analysts define the target segment of a business group. Also, a project that is defined for a business group addresses the business problems for either of the payment modes (postpaid and prepaid). Besides, behavior of customers is inherently different for both the payment modes. Considering all these facts, the analytical model for cross-sell and up-sell has to be built for only one business group. For details about defining business groups, projects, and target segments, see *SAS Offer Optimization for Communications: User's Guide*.

However, if you want to use cross-sell and up-sell as a stand-alone module, then you can use the subset criterion feature of DABT in order to select the population for the cross-sell and up-sell model. For details, see *Chapter 4 Working with Dynamic ABT* in this guide.

Modeling Level

A cross-sell and up-sell model can be built at customer or subscription level.

When a communications service provider campaigns for cross-sell and up-sell of offers, customers decide whether to accept or reject the offers that are recommended to them. Therefore, for cross-sell or up-sell of offers, the level of the model will be customer. The cross-sell and up-sell scores for offers are produced at customer level. These scores are not aggregated further.

Conversely, a service is activated for a subscription and not for a customer. Therefore, for cross-sell or up-sell of services, the level of the model will be subscription. The

variables that are defined at customer level are also used for all subscriptions of a customer. The modeling or scoring process produces cross-sell and up-sell propensity scores for every subscription or customer in the ABT. Therefore, the cross-sell and up-sell scores for services are produced at subscription level. The scores that are produced at subscription level are aggregated to customer level. Scores are aggregated by using weighted means.

Historical Data

Cross-sell and up-sell considers monthly historical data for postpaid customers and requires at least 8 months of data. Conversely, this component considers weekly data and requires at least 6-12 weeks of data for prepaid customers.

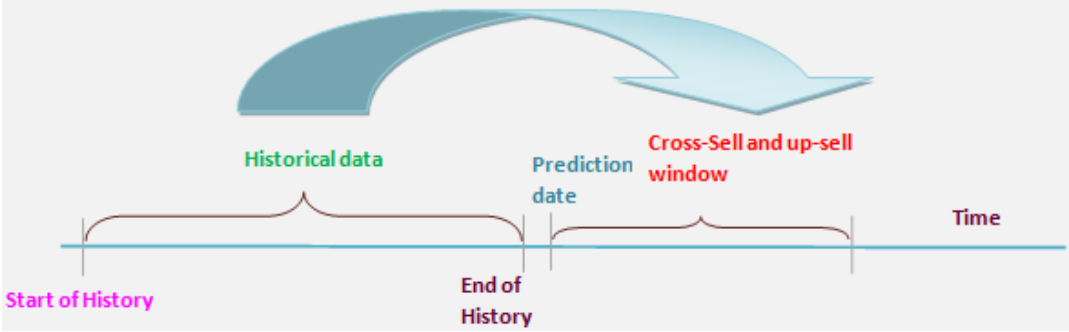
Time Grain

The inherently different nature of business for prepaid and postpaid requires different processing time grains. Therefore, for postpaid payment mode, monthly aggregation of variables is to be used. For prepaid payment mode, weekly aggregation of variables is to be used. All the cross-sell and up-sell score predictions for postpaid customers is at monthly frequency; for prepaid customers, and the predictions are at weekly frequency.

Time Windows Definitions

The following diagram illustrates the time windows that are recommended for the cross-sell and up-sell model.

Figure 7.2 *Time Windows for Cross-Sell and Up-Sell*



Historical data window
: the period for which data is collected and aggregated. Data that is used for prediction (subscriber behavior, payments, and so on) should belong to this period. In other words, data that is generated before or after this period cannot be considered for prediction. The length of the historical time window is determined by the time periods that are defined for the behavioral variables in DABT. For details, see the Behavioral worksheet in "About the ABT-Specific Workbook" on page 61.

Cross-sell and up-sell window
: the period in which a cross-sell or up-sell event is predicted. Typically, a service is activated or an offer is accepted during this period. This window is also called the scoring window. The length of the scoring window depends on the target period count that is associated with the ABT. For details, see the ABT worksheet in"About the ABT-Specific Workbook" on page 61.

Prediction date
: the first day after the historical data period.

Building an Analytical Model

Overview

In model building mode, the analytical user draws a sample of the entire communications service provider's data and builds an analytical model on the sample data. The model learns different behavioral patterns in the sample data. After the user is satisfied with the results that the model produces, it is then registered in the metadata and used in the scoring mode. The sample that is drawn for building a model in the modeling run has to be representative of data that is required in the scoring run.

Defining the Target Variable

Cross-sell and up-sell supports two types of scenarios depending on whether you are creating models for a service or an offer. There can be different criteria that mark a cross-sell or up-sell of a service or an offer.

Example 1

Defining a target variable can be activating a service when it is offered to a customer. The target variable is marked as 1 if the offered service was accepted or activated by the customer within the cross-sell and up-sell time window. The variable is marked as 0 if the customer did not activate the service. However, because the activation is at subscription level, the modeling level will also be at subscription level.

Example 2

Acceptance of an offer by a customer can be considered as cross-sell or up-sell. If a customer accepts the offer, then the xup_flg value is set to 1. Otherwise, it is set to 0.

As explained in example 1, the target variable can be defined as follows:

```
IF HISTORY_END_DT <= SERVICE_ACTIVATION_DT <= (HISTORY_END_DT + TARGET_PERIOD_CNT) THEN
XUP_FLG=1
ELSE
XUP_FLG=0
```

As explained in example 2, the target variable can be defined as follows:

```
IF HISTORY_END_DT <= OFFER_AGRMNT_START_DT <= (HISTORY_END_DT + TARGET_PERIOD_CNT) THEN
XUP_FLG=1
ELSE
XUP_FLG=0
```

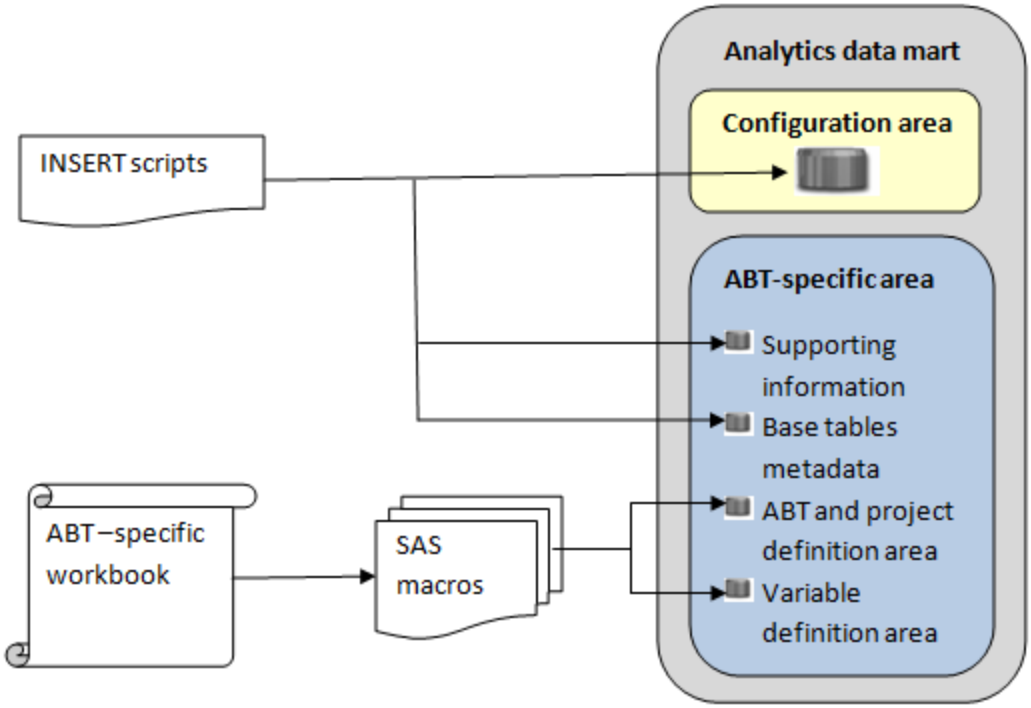
Here, HISTORY_END_DT indicates the end of the historical data window, TARGET_PERIOD_CNT signifies the length of the scoring window, and XUP_FLG is the name of the target variable.

Analytics Data Mart Population

Defining Modeling ABT

The Analytics data mart is categorized into a configuration area andan ABT-specific area. For details, see *Chapter 4 Working With Dynamic ABT* of this guide.

Figure 7.3 *Data Population in Analytics Data Mart*



After the ABT definition and the variables that are part of the ABT are defined, they are stored in the ABT-specific area of the Analytics data mart. Based on this ABT-specific information, DABT extracts the relevant data from the base tables for analytics. The ABT is then built according to the ABT definition that is stored in the Analytics data mart.

After the required information is configured in the configuration area of the Analytics data mart, the analytical modeler can define the ABT. An ABT is defined for a particular purpose in order to solve a particular business problem such as cross-sell and up-sell. Defining an ABT involves populating the details about subset query, the DABT project, and the ABT itself in the ABT-specific workbooks. For details, see "Populating the ABT-Specific Area of Analytics Data Mart" on page 61. In order to store the ABT definition in the Analytics data mart, you run certain macros, in the specified sequence. For details, see "Creating ABTs for Cross-Sell and Up-Sell" on page 132.

Defining ABT Variables

After the ABT definition is populated in the Analytics data mart, the detailed information about the variables that will be a part of that ABT also need to be populated in the mart. DABT supports four types of variable definitions:recent, supplementary, behavioral, and derived. Details about all the variables that are needed for cross-sell and up-sell have to be filled in the partly prepackaged solution-specific workbook. For details, see "Populating the ABT-Specific Area of Analytics Data Mart" on page 61. After you define the variables in the workbook, you have to run certain SAS macros in sequence in order to save all the variable definitions in the Analytics data mart. For details, see "Creating ABTs for Cross-Sell and Up-Sell" on page 132.

ABT Construction

After the ABT definition and its variable definitions are populated in the Analytics data mart, you can construct the physical ABT. As a part of the ABT building process, based on the variable definitions, appropriate data is extracted from the base tables. The data is then aggregated to the appropriate level as mentioned in the ABT definition, and the ABT is physically created.

The %build_abt macro is used for ABT building. For details, see "Creating ABTs for Cross-Sell and Up-Sell" on page 132.

Model Building and Model Writeback

Overview

After the modeling ABT is built, you can proceed to build the analytical model. In this stage of the process flow, you have to perform the following tasks:

Model building
: includes building and registering the analytical model in the SAS Enterprise Miner environment.

Model writeback
: includes writing the model details back to the Foundation data mart. An extension node that is packaged as a part of the cross-sell and up-sell solution writes the model information to the Foundation data mart.

About the Extension Node

The extension node supports the following functionality:

writeback
: The extension node writes back the following information to the Foundation data mart:

  - analytical model
  - association between the analytical model and the offer or service for which the model is built.

  Also, the extension node writes back the following information to the Analytics data mart:

  - scoring ABT
  - scoring project
  - significant variables of the analytical model against the scoring ABT
  - analytical model that is associated with the modeling ABT and scoring ABT

score code generation
: The extension node generates the score code as a optimizedcode.sas file in a specific location.

Assumptions of Model Building and the Extension Node

The extension node for the cross-sell and up-sell models is based on the following assumptions.

- A model ID contains the metadata ID for the SAS Enterprise Miner project through which the model is being built.
- All errors are tracked either through pop-up messages that are displayed in SAS Enterprise Miner or through the session log. Therefore, error handling is not separately performed for the extension node.
- A diagram in SAS Enterprise Miner has only one writeback node, which is preceded by a score node.
- For a modeling ABT, users can build multiple models in SAS Enterprise Miner. However, after the models are assessed, users can register only a single model. As a result, only one model is in production and only one set of significant variables are available for a certain modeling ABT.
- The extension node should be run only when the user completes all activities in SAS Enterprise Miner.

  Note:

  - Running the extension node multiple times can cause the addition of multiple model-related entries in the Foundation data mart.
  - The changes that are made to the analytical models are not stored in the Foundation data mart. Whenever a model is refreshed, it is considered as a new model. Therefore, the record of the old model is marked as closed and a new record is inserted for the new model.
  - Whenever a model is refreshed, the corresponding score code is overwritten.

Scores Writeback

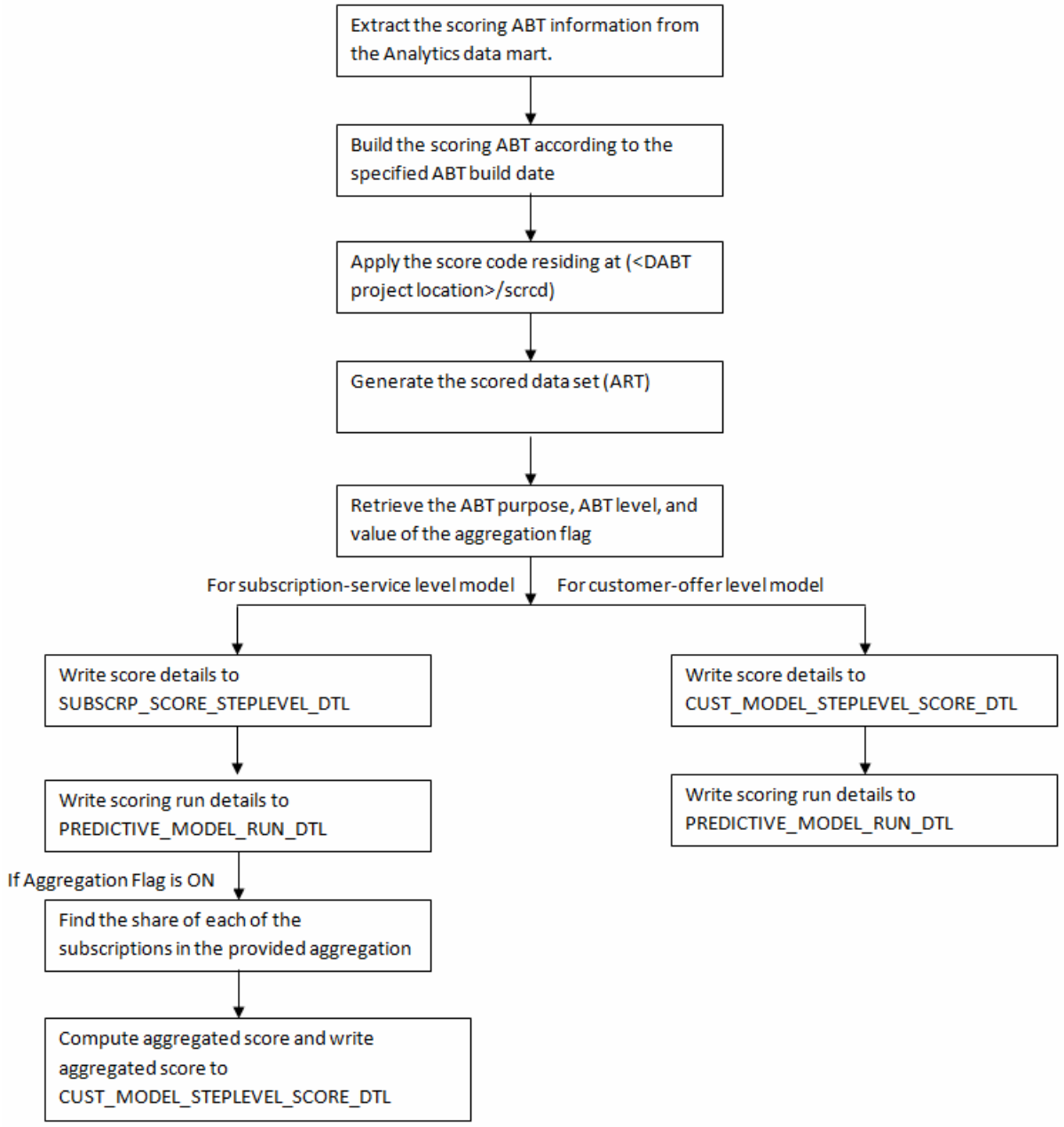
Overview

The scores writeback (also called batch run) assumes that the corresponding modeling ABT is created, models are built, and score codes are registered.

The following diagram illustrates the sequence of tasks that are performed during the scores writeback:

1. scoring ABT generation
2. execution of score code on scoring ABT. In this step, the scoring code is run on the scoring ABT that is generated in the previous step.
3. scores writeback to the appropriate Foundation data mart tables.

Figure 7.4 *Scores Writeback for Cross-Sell and Up-Sell*



Scoring ABT Generation

The scoring ABT is created based on the scoring ABT information that the extension node registers in the Analytics data mart. The scoring ABT is created only for the significant variables of the model.

Score Code Application

The scoring ABT that is generated in the scoring ABT generation step is the input for the score code application. The score code for every model is stored in the project workspace of the corresponding DABT project that builds the modeling ABT.

Scores Writeback

The scores for all customers and subscriptions are written back to the Foundation data mart. In this step, the data from the scored ABT is read and scores and model information is written in the Foundation mart tables such as CUST_MODEL_STEPLEVEL_SCORE_DTL, SUBSCRP_SCORE_STEPLEVEL_DTL, and PREDICTIVE_MODEL_RUN_DTL.

The scores that are generated at subscription level can be aggregated to the customer level. The scores are aggregated based on certain rules.

The usage revenue at subscription level is used for assigning scores to customer level. For example, a customer has three subscriptions. The subscriptions are generating revenue worth 15, 85, and 25 USD per month respectively, and the cross-sell or up-sell scores at subscription level are 0.35, 0.85, and 0.25 respectively. The cross-sell and up-sell score at customer level can be derived by using the weighted average method.

The formula to generate the cross-sell and up-sell score using the weighted average method is as follows: Customer Level Score = SUM(Subscription score * Monthly subscription revenue) for all subscriptions of the customer / SUM(Monthly subscription revenue) for all subscriptions of the customer.

Therefore, for the above example, the cross-sell and up-sell score will be calculated as follows: CUST_MODEL_SCORE_NUM = (15*0.35+0.8*85+0.45*25)/125 = 0.676 ~ 0.68

Assumptions

The batch run is based on the following assumptions:

- Modeling and scoring ABTs reside in the same folder. The create_project_workspace macro will provide the PROJECT_LOCATION to the DABT macro, which will write it to the PROJECT_MASTER table of the Analytics data mart. This location would be the same for modeling as well as scoring the ABT project. Therefore, this location will then be used by the scoring process to determine the location of the score code.
- Every time the customers or subscriptions are scored, the scores are written back to the Foundation data mart against the scoring date. However, the scoring ABT is overwritten every time the customers or subscriptions are to be scored.
- A scoring project cannot be scheduled multiple times in a day.

Checklist for the Analytical Process Flow

Analytical modelers can refer to the following checklist before they start building analytical models for cross-sell and up-sell.

Table 7.1 *Checklist for Analytical Process Flow*

Process name	Flow	Customer-Offer Level Cross-Sell and Up-Sell	Subscription-Service Level Cross-Sell and Up-Sell
Business Problem Definition	Objective	To provide the probability of accepting an offer when it is offered to a customer.	To provide the probability of accepting a service when it is offered to a customer.
Population Definition	Assumptions	The population can be customers who belong to a business group.	The population can be subscriptions that belong to a business group.
	History	6-8 months for postpaid customers	6-8 months for postpaid customers
		6-12 weeks for prepaid customers	6-12 weeks for prepaid customers
	Level	Customer - Offer	Subscription - Service
	Time Grain	monthly for postpaid customers	monthly for postpaid customers
		weekly for prepaid customers	weekly for prepaid customers
	Time Window Definition	contains historical data window and cross-sell and up-sell window	contains historical data window and cross-sell and up-sell window
	Sampling	defined by using SAS Enterprise Miner	defined by using SAS Enterprise Miner
	Subsetting	defined by using the DABT subsetting criteria. Using this feature, you can select customers based on certain attributes such as tenure with service provider and customer status.	defined by using the DABT subsetting criteria. Using this feature, you can select subscriptions based on certain attributes such as tenure with service provider and subscription status.

Process name	Flow	Customer-Offer Level Cross-Sell and Up-Sell	Subscription-Service Level Cross-Sell and Up-Sell
ABT Definition	Impact of input controls	The DABT project refers to a particular offer to model for. Therefore, the number of DABT projects is equal to the number of offers to model for. Also, the number of ABTs that will be created will be the same.	The DABT project refers to a particular service to model for. Therefore, the number of DABT projects is equal to the number of services to model for. Also, the number of ABTs that will be created will be the same.
	Input Variables	Input variables will be defined based on the underlying business problem and the offer that is to be offered. The input variables will be derived through DABT.	Input variables will be defined based on the underlying business problem and the service that is to be offered. The input variables will be derived through DABT.
Model Building	Target Variable Definition	For customer-offer level modeling, the decision whether the customer opted for the target offer in the target period is important. Therefore, the target variable is marked as 1 if the customer signed up for the offer during the target period and 0 otherwise.	For subscription-service level modeling, the decision whether the customer opted for the target service in the target period is important. Therefore, the target variable is marked as 1 if the customer signed up for the subscription during the target period and 0 otherwise.
	Partitioning	Users can partition the data set in SAS Enterprise Miner by using the standard data set partitioning technique.	Users can partition the data set in SAS Enterprise Miner by using the standard data set partitioning technique.
	Number of Models	Multiple models can be built on one modeling ABT. However, only one of the models will be registered.	Multiple models can be built on one modeling ABT. However, only one of the models will be registered.
	Model Writeback	Model writeback is through an extension node of SAS Enterprise Miner. This node writes back information to Foundation data mart tables.	Model writeback is through an extension node of SAS Enterprise Miner. This node writes back information to the Foundation data mart tables.
	Result Interpretation	The scores will be probability of activating an offer when it is offered to a customer.	The scores will be probability of activating a service when it is offered to a customer.

Process name	Flow	Customer-Offer Level Cross-Sell and Up-Sell	Subscription-Service Level Cross-Sell and Up-Sell
Scoring	Scoring Frequency	Monthly for Postpaid	Monthly for Postpaid
		Weekly for Prepaid	Weekly for Prepaid
	Mechanism	Through batch run macro.	Through batch run macro.
	Score Aggregation	Aggregation is not required as the level is Customer.	Scores are aggregated to the customer level if the aggregation flag is set to Y.
	Scores Writeback	Scores are written back to the Foundation data mart through the batch run macro. This process will be run after score code execution on scoring ABT is complete. This process will write the scores information to the tables such as CUST_MODEL_STEPLEVEL_SCORE_DTL and PREDICTIVE_MODEL_RUN_DTL.	Scores are written back to the Foundation data mart through the batch run macro. This process will be run after score code execution on scoring ABT is complete. This process will write the scores information to the tables such as CUST_MODEL_STEPLEVEL_SCORE_DTL, SUBSCRP_SCORE_STEPLEVEL_DTL, and PREDICTIVE_MODEL_RUN_DTL.

Chapter 8

Working With Cross-Sell and Up-Sell

Configuring the Setup for Cross-Sell and Up-Sell

Overview

Before you build the analytical models for cross-sell and up-sell, you have to complete the following tasks.

Initialize Parameters

The analytical models for cross-sell and up-sell require a set of parameters. In order to configure these parameters, you have to run the cfd_analytics_param_declaration_job job. The cfd_analytics_param_declaration_job initializes the parameters that are required for cross-sell and up-sell models. This job refers to the CFDN_ANALYTICS_CONFIG_PARAM table that resides in the CFDCONF library. The CFDN_ANALYTICS_CONFIG_PARAM table contains names and values of the global parameters. For details, see "Parameters for Customer Analytics" on page 252.

After you run the cfd_analytics_param_declaration_job job, the cfdn_initialization_crcs.sas parameter file is generated.

To initialize the parameters:

1. Verify the values of all the parameters in the CFDN_ANALYTICS_CONFIG_PARAM table and modify the values if necessary.
2. Log on to SAS Data Integration Studio and connect to a profile.
3. On the **Folders** tab, expand **Products** ⇨ **SAS Offer Optimization for Communications** ⇨ **Jobs** ⇨ **Analytics Setup Job Group**.
4. Run the cfd_analytics_param_declaration_job job.
5. Close SAS Data Integration Studio.
6. Go to the **`<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode`** folder.
7. Verify that the cfdn_initialization_crcs.sas initialization file is created at the specified location.

Populate ABT Data for Sample Model

The sample model that is packaged will use the prepackaged ABT data. For cross-sell up-sell, two sample models are provided, one at customer-offer level and another at subscription-service level. You have to update and run INSERT scripts for ABT data population.

To update and run INSERT scripts:

1. Create **`smplcustoffr`** and **`smplsubssrvc`** folders for cross-sell and up-sell at offer and service levels respectively, in the following folder: **`<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu`**.
2. Create the **`abt`** folder in the following folders:

- **For customer-offer level model:** **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/smplcustoffr**
- **For subscription-service model:** **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/smplsubssrvc**

3. Go to the **<SAS configuration path>/Lev1/<SAS Application Server context name>** folder.
4. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
5. Declare the ABT library in Base SAS to point to the following locations depending on whether you are building the model for customer-offer level cross-sell and up-sell model or subscription-service level cross-sell and up-sell model:
   - **For customer-offer level model:<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/smplcustoffr/abt**
   - **For subscription-service level model:** **<SAS configuration ath>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/smplsubssrvc/abt**

   Note: Perform steps from 6 to 11 for customer-offer level model.
6. Open the ca_insert_for_abt_xsus_custoffr.sas macro. This macro is located in the following folder: **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts**
7. Verify the macro arguments.

   ```
   %ca_insert_for_abt_xsus_custoffr (cfdn_cslib_nm=abt, cfdn_csabt_nm=abt_xsus_cus
   toffr);
   ```

8. Enter appropriate values for the macro arguments.
   a. Specify the abt value for the cfdn_cslib_nm argument.
   b. Specify the abt_xsus_custoffr value for the cfdn_csabt_nm argument.
9. Click **Save**.
10. Click **Run**.
11. To ensure successful execution of the INSERT scripts, view the log.

    Note: Perform steps 12 to 17 for subscription-service level model.
12. Open the ca_insert_for_abt_xsus_subsrvc.sas macro. This macro is located in the**<SAS installation path>/SASFoundation/9.2/comfdnsrv/sasmisc/Controlscripts/insertscripts** folder.
13. Verify the macro arguments.

    ```
    %ca_insert_for_abt_xsus_subsrvc (cfdn_cslib_nm=abt, cfdn_csabt_nm= abt_xsus_subscrpsrv);
    ```

14. Enter appropriate values for the macro arguments.
    a. Specify the abt value for the cfdn_cslib_nm argument.

b. Specify the abt_xsus_subscrpsrv value, for the cfdn_csabt_nm argument.

15. Click **Save**.
16. Click **Run**.
17. To ensure successful execution of the INSERT scripts, on the menu select **View** ⇨ **Log**.
18. Close Base SAS.

Import Sample Models

Sample models for cross-sell up-sell are prepackaged with SAS Offer Optimization for Communications. A sample model is defined for each level (customer-offer level and subscription-service level). The purpose of these models is to demonstrate their analytical flow in SAS Enterprise Miner. Also, the extension node that is packaged for these models populates the tables that are defined for CFDDIM, CFDFACT, and DABT_ADM libraries.

Note: Make sure that you do not run the extension node (also called writeback node) of the sample model.

The sample cross-sell and up-sell model for customer-offer level is packaged as xsus_custoffr_sampl_pkg.spk; the sample cross-sell and up-sell model for subscription-service level is packaged as xsus_subscrpsrv_sampl_pkg.spk.

To install and configure the sample models:

Install Sample Models

1. Start the SAS Analytics Platform server.
2. Log on to SAS Enterprise Miner and connect to the appropriate server.
3. On the menu, select **File** ⇨ **New** ⇨ **Project**. The Create New Project dialog box appears.
4. Select the appropriate server and click **Next**.
5. In the **Project Name** box, type *xu_custoffr_model* or *xu_suscrpsrv_model* depending on whether you are importing the model for customer-offer or subscription-service level and click **Next**.
6. In the **SAS Server Directory** box, type the appropriate path for the customer-offer level model as *<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/smplcustoffr/model* or for the subscription-service level model as *<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/smplsubssrvc/model* respectively and click **OK**.
7. Do not change the default values for the remaining steps and click **Next** until you reach the last screen.
8. Click **Finish**. In the SAS Enterprise Miner window, the new project appears at the top left of the **Project** panel.

Configure Sample Models

9. Select the project and then select the project start-up code from the property column. The Project Start Code window appears.
10. To use the sample model with the sample data sets, define the library that contains the sample data sets.

a. Select the Project Start Code window.

b. Type the following code for the customer-offer level cross-sell and up-sell model:

```
libname abt <SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2
/Data/common_data/analytics/xu/smplcustoffr/abt;
```

Type the following code for subscription-service level cross-sell and up-sell model:

```
libname abt <SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2
/Data/common_data/analytics/xu/smplsubssrvc/abt;
```

c. Click **Run Now**.

d. Click **OK**.

11. Right-click **Data Source** and in the Property window, select **Create Data Source**.

12. In the Data Source Wizard, complete these steps:

a. Select the default value (SAS Table) for the metadata source and click **Next**.

b. Click **Browse** to select a SAS table. All the predefined SAS libraries are displayed.

c. Select the ABT library depending on whether the ABT built is for customer-offer level or subscription-service level.

- **For customer-offer level model:** abt_xsus_custoffr
- **For subscription-service level model:**abt_xsus_subscrpsrv

d. Click **OK**.

e. Click **Next**. The table properties window appears.

f. Click **Next**.

g. Select **`Advanced`** from the metadata advisor options and click **Next**. The details of variables included in the ABT are displayed.

h. Set the role of the variables as explained below:

Note: The following tables indicate the roles of only the important variables. For the other variables, the role can be ID or Input.

Table 8.1 *Roles for Subscription-Service Level XSUS ABT Variables*

Variable name	Role
XUPR_FLG_SS_S9	Target
AVG_REV_AMT	Input

Table 8.2 *Roles for Customer-Offer Level XSUS ABT Variables*

Variable name	Role
XUP_O14_FLG	Target

i. In the Decision Configuration page, select **No**, and then click **Next**.

j. Click **Next**. The information about the ABT variables is displayed.

k. Click **Finish**.

13. On the menu, select **File** ⇨ **Open Model Package**.

14. Click **Browse** and locate the xsus_custoffr_sampl_pkg.spk file. On a Windows machine, this file is available in the **<SAS installation path>/SASFoundation/bppsrv/sasmisc/sampleeminermodels/xsusmodelcustoffr** path. However, for a UNIX machine, if the spk file is on another machine, SAS Enterprise Miner must have network access to the file, or you must copy the file to your machine.

15. Select the xsus_custoffr_sampl_pkg.spk file, and then click **Open**. The model diagram is displayed.

16. Right-click the diagram and on the **Actions** menu, select **Recreate Diagram**.

17. In the Input dialog box, type a name for the diagram, and then click **OK**.

18. Click **Yes** in the Confirmation dialog box. The model diagram is displayed.

19. Right-click the **Diagrams** option in the project tree and select **Save as** from the menu. Select a suitable location to save the diagram.

20. Type the filename as *xsus_custoffr_diag*. Make sure that the file type is **xml**. This diagram will be used when you create an offer-level cross-sell and up-sell model. For details, see "Building a Customer-Offer-Level Cross-Sell and Up-Sell Model" on page 141.

21. Repeat steps 19 and 20 for the subscription-service level sample model. Type the filename for the diagram as *xsus_subsrvc_diag*. This diagram will be used when you create a subscription-level cross-sell and up-sell model. The sample model that is packaged for subscription-service level cross-sell and up-sell model is available at the following location: **<SAS installation path>/SASFoundation/bppsrv/sasmisc/sampleeminermodels/xsusmodelsubssrv**For details, see "Building a Subscription-Service-Level Cross-Sell and Up-Sell Model" on page 144.

Creating ABTs for Cross-Sell and Up-Sell

Prerequisites

Before you create the ABTs for cross-sell and up-sell, make sure that the following tasks are complete:

1. Populate the configuration area of the Analytics data mart with correct data. For details, see "Populating the Configuration Area of Analytics Data Mart" on page 42.

2. Familiarize yourself with the worksheets and the columns in the ABT-specific workbook. For details, see "About the ABT-Specific Workbook" on page 61.

3. Familiarize yourself with the recent behavioral, supplementary, and derived variables by reading through the Analytics data model. For details, see *SAS Offer Optimization for Communications: Analytics Data Mart*.

4. Refer to the code of the cfdn_outer_macro.sas macro. This macro is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/**

`SASOfferOptForCommServer5.2/SASCode/insertscripts` folder. This macro contains the sample calls for all the macros, which need to be run in order to create the ABTs. When you perform the instructions that are explained below, make sure that you have opened the cfdn_outer_macro.sas macro. Also, the instructions specify the sequence in which these macros are to be run. Before you run the macros from this macro, make sure that you verify the argument values that these macros require.

Acquire the Purpose-Specific ABT Configuration File

Depending on the purpose for which you are defining the ABT template, you have to use the correct ABT configuration workbook.

Table 8.3 *Purpose-Specific ABT Configuration Workbooks for Cross-Sell and Up-Sell*

Purpose	Purpose Short Name	Workbook Name
Cross-sell and up-sell customer offer postpaid	XUPOST	ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls
Cross-sell and up-sell customer offer prepaid	XUPRE	ABT_SPECIFIC_CSUSprepaid_CustXOffer.xls
Cross-sell and up-sell subscription service postpaid	XUPOST	ABT_SPECIFIC_CSUSpostpaid_SubsXService.xls
Cross-sell and up-sell subscription service prepaid	XUPRE	ABT_SPECIFIC_CSUSprepaid_SubsXService.xls

Make sure that you have downloaded these workbooks and saved them in an appropriate location on your computer. For details, see "Download the Analytics Workbooks" on page 41.

Note: This section refers to the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls workbook wherever necessary. However, you have to refer to the appropriate workbook according to the purpose for which you are defining the ABT template.

Create User in SAS Management Console

The user who executes the INSERT scripts that are mentioned in the instructions below needs access to the libraries that are predefined in the SAS environment. In order to enable the user to have this capability, make sure that a user is defined in SAS Management Console with the following role and group:

- Role: Metadata Server: Unrestricted
- Group: Communication Common Oracle User Group

Update the Name and Location of the Configuration File

After you have acquired one of the above appropriate workbooks, complete the following steps:

1. Go to the **<SAS configuration path>/Lev1/<SAS Application Server context name>** folder.
2. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
3. Open the cfdn_outer_macro.sas file, which is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts** folder.
4. Add the following code:

   ```
   %let abtxls =<path in which you have stored the purpose-specific
   workbook>
   ```

   For example, you can add the following lines of code if you are working with the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls workbook:

   ```
   %let abtxls =C:/BPP/common/docs/docs_2/final/ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls
   ```

5. Click **Save**.
6. Run the line of code that you have added in step 4.
7. To ensure successful execution of the code, on the menu select **View** ⇨ **Log**.

Define the Population

When you create an analytical model, you have to define the population for which you are building the analytical model. The subset criteria feature of DABT enables you to define your population. The subset conditions might differ depending on the business group definitions.

Note: One selection criterion is associated with only one modeling ABT and, therefore, only one DABT project.

To define the population for your analytical model:

1. Open the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls file.
2. In the **subset_where** worksheet, populate appropriate values for the parameters. For example, the WHERE_CLAUSE_EXPRESSION parameter determines the period for which historical data is required for building the ABTs.
3. Save and close the file.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_SUBSETQUERY_IMPORT
   - %CFDN_CREATE_SUBSET_WRAPPER.

   As a result, subset queries are entered into the SUBSET_QUERY_MASTER table of the Analytics data mart.

Create a DABT Project

You have to define a DABT project in order to build a solution-specific ABT. A DABT project caters to only one modeling ABT.

To create a DABT project:

1. Open the appropriate workbook (for example, the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls file).
2. In the **Project** worksheet, specify the project details such as PROJECT_NM, LEVEL_NM, DESCRIPTION, LOCATION_PATH, Query_Nm, and Purpose.

   Note: When you specify details for Query_Nm, make sure that you refer to the query that you have created in the previous step. For details, see "Define the Population" on page 134.
3. Save and close the file.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_PROJECT_IMPORT
   - %CFDN_NEW_PROJECT_WRAPPER

   The PROJECT_MASTER table of the Analytics data mart is populated with the details of the new project.

Create a Project Workspace

Each project has a workspace in which the modeling and scoring ABTs, configuration details, and score code are stored.

To create a workspace for a project, run the %CFDN_CREATE_PROJECT_WORKSPACE macro. This macro has the argument cfdn_crcs_data_path. Specify the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics** value for this argument. A folder is created in this path. The name of the folder that is created is the same as the name of the DABT project. This will be referred to as <project_nm> in the subsequent paths when you create the project. In addition, subfolders **abt**, **art**, **model**, and **srcd** are created in this folder.

This macro is based on the following assumptions:

- This macro reads values of ABT_NM, PROJECT_NM, and PURPOSE from the d_project temporary table of the **scrflx** library and creates the following folders in the project workspace:
  - abt
  - art
  - scrcd
  - model
- The ABT resides in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/<project_nm>/abt** folder. All modeling and scoring ABTs reside in the same location.
- The SAS Enterprise Miner project name is user-specific. The DABT project is different from the SAS Enterprise Miner project.
- All SAS Enterprise Miner projects that are built on a given modeling ABT reside in the DABT project workspace of a modeling ABT (that is, in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/<project_nm>/model** folder).

For example, assume that the data folder resides in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics** folder. If the ABT_NM is abt_xsus_custoffr, Purpose is XUPOST, and PROJECT_NM is CrossSellProject, then the folder structure is **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/CrossSellProject/model**.

Therefore, a user can build a SAS Enterprise Miner project with this ABT with the name, *cross-sell for postpaid customers*. The SAS Enterprise Miner project workspace is created in the following folder:

<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/CrossSellProject/model/cross sell for postpaid customers

- The project workspace creation macro calls two integration macros. These macros are executed for all DABT projects.

  %cfdn_new_project_specific_lib
  : After you create the project workspace, this macro generates the library name and libref. It also makes an entry of the ABT library for that project in the LIBRARY_MASTER table of the Analytics data mart. In addition, this macro creates an entry against the LOCATION_PATH for that project in the PROJECT_MASTER table of the Analytics data mart.

  %cfdn_create_include_libsfile
  : This macro creates the libs_include.sas file in the **scrcd** folder of the project workspace. This file contains a declaration of the ABT library that is generated when you run the %cfdn_new_project_specific_lib macro.

Update the Library Reference

In this step, the user has to manually input the library short name that gets created in the "Create Project Workspace" step to the modeling ABT to be created. The user has to make this entry in the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls file in the **ABT** worksheet area under the library field.

To update the library reference:

1. Open the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls file.
2. In the **ABT** worksheet, for the library column, enter the library short name that is created when you create a project workspace.
3. Save and close the file.

Define Modeling ABT

This step makes the following assumptions:

- The library reference and the LIBNAME statement that are entered in the **ABT** worksheet should be the same as those entered in the **Project** worksheet of the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls file.
- The libref should point to a location in which the modeling ABT would reside after you create it. This is the ABT folder under a DABT project workspace. That is, it should point to the following location:**<SAS configuration path>/Lev1/**

Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/<project_nm>/abt

To define a modeling ABT:

1. Open theABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls file.
2. In the **ABT** worksheet, provide details such as Table_Nm, Short_Nm, Description, Target_Time_Freq, Target_Time_Freq_Count, Library, LIBRARY_REFERENCE, LIBNAME_STATEMENT, Level, PROJECT_NM, FLG, Purpose, and ABT_TYPE.
3. Save and close the file.
4. In Base SAS, run the following macros from the %cfdn_outer_macros.sas macro.
   - %CFDN_ABT_IMPORT
   - %CFDN_NEW_ABT_WRAPPER

   As a result, a new record of the modeling ABT that is to be created with respect to the DABT project is added in the ABT_MASTER table.

Define ABT Variables

To define variables for the modeling ABT:

1. Open the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls workbook.
2. In the **Supplementary**, **Recent**, and **beh** worksheets, specify appropriate values for the columns. For details, see "About the ABT-Specific Workbook" on page 61.
3. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro.
   a. Run the %cfdn_abtvars_import macro. This macro converts the data from the worksheet to a SAS data set.
   b. Run the %cfdn_abt_dimval_insert macro. This macro creates temporary SAS data sets for each type of variable that is mentioned in the macro.

      Note: Make sure that you run this macro only for Behavioral (BEH) and Recent (RNT) variables.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_CREATE_BEH_VAR_WRAPPER
   - %CFDN_CREATE_RNT_VAR_WRAPPER
   - %CFDN_CREATE_SPM_VAR_WRAPPER

   This step creates data sets for the supplementary, recent, and behavioral variables in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/dabt_data/scratch/flex** folder. These data sets are created in the scrflx library (for example, the beh_var_list_15.sas7bdat and beh_var_list_15_edit.sas7bdat).
5. In all the *_edit.sas7bdat data sets, verify the variable names. If the length of the variable names exceeds the maximum limit, modify the variable names.

   Note: The values for edit_cd and edit_option columns indicate whether the length of the variable names exceeds the maximum limit of 28 characters. Rename all the variables where edit_option is set to D (representing delete as the length is greater than 28 characters). Edit names of all such variables and change the edit_option for those to A to represent "accept" because the length is less than or equal to 28 characters.

6. To save the variables, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_SAVE_BEH_VAR_WRAPPER
   - %CFDN_SAVE_RNT_VAR_WRAPPER
   - %CFDN_SAVE_SPM_VAR_WRAPPER

   This step reads data from the temporary tables and enters data in the following tables:
   - VARIABLE_MASTER
   - SUPPLEMENTARY_VAIABLE
   - RECENT_VARIABLE
   - BEHAVIORAL_VARIABLE
   - ABT_X_VARIABLE
   - VARIABLE_DIM_ATTRIBUTE_FILTER

Define Derived Variables

Before you create derived variables, make sure that you have complete information about them. For details, see "Derived Variables" on page 259.

To create derived variables:

1. Open the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls workbook.
2. In the **derivedvardetails** and **derivedvarsource** worksheets, specify appropriate parameter values.
3. Save and close the file.
4. In Base SAS, run the following macros from the cfdn_outer_macro.sas macro:
   - %CFDN_DRVDVARS_IMPORT
   - %CFDN_SAVE_DRVD_VAR_WRAPPER

Define Target Variable

A target variable is defined only for modeling ABTs. DABT enables you to define the target variable according to your requirements. A target variable is a derived variable. You can define the target variable through the **derivedvardetails** and **derivedvarsource** worksheets of the corresponding ABT_SPECIFIC.xls file.

When you build a modeling ABT for cross-sell up-sell, customers are observed for the service activations or offer acceptance. These activations are observed for a certain period from end of history. This period is called the target period. A TARGET_PERIOD_CNT is associated with ABT at the time of definition. This period indicates the duration of the target period. The frequency of the target period can be month, week, or day. A target period frequency is also associated with the ABT at the time of definition. For details, see the **ABT** worksheet in "About the ABT-Specific Workbook" on page 61.

To define a target variable:

1. Open the ABT_SPECIFIC_CSUSpostpaid_CustXOffer.xls sheet.

2. In the **derivedvardetails** and **derivedvarsource** worksheets, specify appropriate parameter values.
3. Save and close the file.
4. Find the sk of the ABT that you are building and the sk of the target variable in that ABT. You can do so by searching for the physical name of the variable and the ABT in the VARIABLE_MASTER and ABT_MASTER tables.
5. In Base SAS, run the %outcome_var macro from the %cfdn_outer_macro.sas macro. This macro sets the value of the PART_OF_ABT_FLG column in the ABT_X_VARIABLE table to Y. For example, a sample call to this macro can be as follows:

   ```
   %outcome_var(m_abt_sk=1, m_abt_var_sk=108);
   ```

   Note: If a variable is chosen as outcome variable, then it must be a part of that ABT. In other words, the part_of_abt_flg must be set to Y for that variable. This indicates that unless a variable is marked to be computed for an ABT, it cannot be marked as an outcome variable.

Select ABT Variables

To select the variables that are to be included in cross-sell and up-sell ABTs, the editable data set corresponding to the ABT_X_VARIABLE table and to the ABT under consideration is first created. The ABT_X_VARIABLE_edit_* data set is created in the scrflx library. The part_of_abt_flg in this data set can be set to Y or N depending on whether the corresponding variable is to be included in the ABT.

After the data set is edited, the changes that are made have to be written back to the ABT_X_VARIABLE data set.

To complete the above mentioned tasks:

1. Identify the name and the SK of the ABT that you are working with.
2. Identify variables that are to be excluded or included in the ABT and make a list of their SKs.
3. In Base SAS, run the % cfdn_create_edit_abt_x_var macro from the %cfdn_outer_macro.sas macro. This macro creates the ABT_X_VARIABLE_edit_* data set corresponding to the ABT_X_VARIABLE table of the scrflx library.
4. In Base SAS, open the ABT_X_VARIABLE_edit_* data set.
5. Set the value of part_of_abt_flg to Y or N according to your requirement.
6. In Base SAS, run the %CFDN_UPDATE_ABT_X_VAR_LIST macro from the %cfdn_outer_macro.sas to write these changes to DABT_ADM.ABT_X_VARIABLE table.

Construct ABT

You have to run the %build_abt macro for building the ABT. Specify the ABT_SK of the ABT that is to be constructed and the date on which the ABT is constructed in the macro call.

To construct the ABT:

1. In Base SAS, open the build_abt macro. The call for this macro is in the cfdn_outer_macro macro.
2. Specify appropriate values for the following arguments. These arguments are a part of a deployed ETL job that the build_abt macro calls.

   bpp_prj_tmp_data_dir
   path that is specific to the SAS Offer Optimization for Communications application and is not used for customer analytics. In the autoexec file, you can set the value for this parameter as NULL. The application dynamically sets the value as required.

   bpp_prj_abt_dir
   path that is specific to the SAS Offer Optimization for Communications application and is not used for customer analytics. In the autoexec file, you can set the value for this parameter as NULL. The application dynamically sets the value as required.

   trg_prd
   value for the target period count for the ABT, which is to be built.

   bg_id
   ID for the business group that is used in the subsetting criterion.

   m_abt_sk
   surrogate key of the ABT that you want to build.

   m_abt_bld_dt
   the date on which the ABT needs to be built. The historical data that will be used for computing variables will belong to the period that ends a day before this date.
3. Run the %build_abt macro. For example, the code can be as mentioned below.

```
option mlogic mprint symbolgen;
%let bpp_prj_tmp_data_dir =C:\SAS\Config\Lev1\Applications\SASOfferOptForCommServer5.2
\Logs\dabt_logs\flex;
%let  bpp_prj_abt_dir =C:\SAS\Config\Lev1\Applications\SASOfferOptForCommServer5.2
\Logs\dabt_logs\flex;
%let bg_id=9_BUSINESS_GROUP_ID;
%let trg_prd = 1;
%build_abt(m_abt_sk = 31, m_abt_bld_dt=20APR2009);
```

Example: Building a Modeling ABT for Cross-Sell and Up-Sell

Assume that the ABT_SK is 10, ABT name is sample_abt, the corresponding DABT project name is sample_prj, and the purpose of the ABT is cross-sell and up-sell. After you run the %build_abt macro, the following steps are executed:

- The high-level logs, build_abt_10 and build_subset_criteria_10, are created at the location: **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Logs/dabt_logs/flex**.
- The modeling ABT is created in the following folder: **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/sample_prj/abt** .
- The intermediate data sets are created in the following folder:**<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/dabt_data/scratch/cm/10**. This location will be accessible in the Base SAS session as a SAS library, named work_area.

- The PARAMETER_LIST_HM data set that is created at the above location contains the error codes for all the steps that are executed in order to build the ABT. The error codes are stored in the RETURN_CD column of the PARAMETER_LIST_HM data set. The value of the RETURN_CD column corresponding to each JOB_NM column should be 0 for building the ABT successfully.
- The logs corresponding to values of all JOB_NM columns for the PARAMETER_LIST_HM data set are stored in the following folder: **`SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Logs/dabt_logs/schedule_cm/sample_abt`**.

Building a Customer-Offer-Level Cross-Sell and Up-Sell Model

Prerequisites

Before creating a cross-sell and up-sell model, make sure that a suitable user ID is defined for performing tasks in SAS Enterprise Miner. For details, see *SAS Offer Optimization for Communications: Installation and Configuration Guide*.

Also, ensure that libraries DABT_ADM, CFDDIM, CFDFACT, and CFDMISC are available in the SAS Enterprise Miner environment. If these libraries are not available, then declare them in the SAS Enterprise Miner start-up code.

In addition, make sure that a modeling ABT is created through DABT, and the DABT project workspace exists. For details, see "Create a DABT Project" on page 94.

The following sections refer to the DABT cross-sell and up-sell project as **`xsus_custoffr_project`**.

Create a Customer-Offer-Level Cross-Sell and Up-Sell Model in SAS Enterprise Miner

Log on to SAS Enterprise Miner

To create a cross-sell and up-sell model, log on to SAS Enterprise Miner with a profile and connect to a server.

Create a Project

1. Click **New Project**.
2. In the Create New Project wizard, complete these steps:
   a. Select the **SAS Server** and click **Next**.
   b. In the **Project Name** box, type *Cross-Sell and Up-Sell Model for Postpaid Customers*.
   c. In the **SAS Server Directory** box, enter the path for the **`model`** folder that is created in the DABT project workspace. For example, you can type the path as **`<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_custoffr_project/model`**.

d. Click **Next**.

e. Do not change the default folder location that is displayed for **SAS Folder Location** and click **Next**.

f. Click **Finish**. This step creates the SAS Enterprise Miner project and also the project workspace.

3. Select the *Cross-Sell and Up-Sell Model for Postpaid Customers* project and then in the property section, click **Project Start Code**.

4. In the Project Start Code window, copy the code from the cfdn_xsus_em_startup.sas file. This code is available in the **`<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommunication5.2/SASCode/insertscripts`** folder.

   Note: Make sure that you enter proper values for all the parameters in the start-up code. Whenever the SAS Enterprise Miner model details are being written back for the first time, the value of cfdn_reexec_flg should be N. For all the subsequent runs, the value of cfdn_reexec_flg should be Y.

5. Click **Run Now**.

6. Click **Log** to view the log file and make sure that the log file does not contain any errors.

7. Click **OK**.

Create Diagram

1. In the project tree, right-click **Diagrams** and select **Import Diagram from xml**.

2. Select the **`xsus_custoffr_diag.xml`** that you created when you imported the sample model.

3. Click **Open**. The diagram is imported and opened in the Diagram pane.

Create Data Source

1. Right-click **Data Source** and in the Property window, select **Create Data Source**.

2. In the Data Source Wizard, complete these steps:

   a. Select the default value (SAS Table) for the metadata source and click **Next**.

   b. Click **Browse** to select a SAS table. All the predefined SAS libraries are displayed.

   c. Double-click the library in which you stored the modeling ABT for customer-offer level cross-sell and up-sell and select the appropriate data set. For details, see "Construct ABT" on page 139. For example, the data set can be xsus_custoffr_data. The subsequent steps refer to this data set.

   d. Click **OK**.

   e. Click **Next**. The table properties window appears.

   f. Click **Next**.

   g. Select **`Advanced`** from the metadata advisor options and click **Next**. The details of variables included in the ABT are displayed.

   h. Set the role of the XUP_OFFR_FLG variable as Target.

      Note: For other variables, the role can be ID or Input. Make sure that the level of the target variable is binary.

i. In the Decision Configuration page, select **No**, and then click **Next**.

j. Click **Next**. The information about the ABT variables is displayed.

k. Click **Finish**.

Build Customer-Offer-Level Cross-Sell and Up-Sell Model

1. Delete the input data set of the diagram that was created in the Create Diagram step.
2. Drag and drop the xsus_custoffr_data data set in the **Diagram** pane and connect it to the first node of the diagram.
3. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Train`**.
4. Delete the input data set of the diagram that is connected to the score node.
5. Drag and drop the xsus_custoffr_data data set in the **Diagram** pane and connect it to the score node of the diagram.
6. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Score`**.
7. Right-click the xsus_custoffr_data data set, which is connected to score node and click **Edit Variables**.
8. Set the role of the XUP_OFFR_FLG variable to **`Rejected`**.
9. Right-click the **Model Writeback** node and select **Update**.
10. After the diagram is updated successfully, right-click the **Model Writeback** node and select **Run**. Select **Yes** when asked for confirmation.
11. Click **OK** when the run is complete. In this step, the following tasks are completed:
    - The ANALYTICAL_MODEL_D table of the Foundation data mart and the ANALYTICAL_MODEL_MASTER table of the Analytics data mart are updated.
    - The association between the model and the offer for which the model is being built is stored in the ANALYTICAL_MODEL_DTL table of the Foundation data mart.
    - The optimizedcode.sas score code is generated in the **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_custoffr_project/scrcd`** folder.
    - The significant variables of the model are written in the abt_sgnfcnt_vars data set. This data set is available in the following location: **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_custoffr_project/scrcd`**
    - The xsus_custoffr_project_scr scoring project is registered for the xsus_custoffr_data modeling ABT in the PROJECT_MASTER table of the Analytics data mart.
    - In the batch run, the xsus_custoffr_data_scr scoring ABT is generated in the following location: **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_custoffr_project/abt`**.
    - The significant variables are added in the ABT_X_VARIABLE table of the Analytics data mart against the scoring ABT (xsus_custoffr_data_scr).

Building a Subscription-Service-Level Cross-Sell and Up-Sell Model

Prerequisites

Before creating a cross-sell and up-sell model, make sure that a suitable user ID is defined for performing tasks in SAS Enterprise Miner. For details, see *SAS Offer Optimization for Communications: Installation and Configuration Guide*.

Also, ensure that libraries DABT_ADM, CFDDIM, CFDFACT, and CFDMISC are available in the SAS Enterprise Miner environment. If these libraries are not available, then declare them in the SAS Enterprise Miner start-up code.

In addition, make sure that a modeling ABT is created through DABT, and the DABT project workspace exists. For details, see "Create a DABT Project" on page 94.

The following sections refer to the DABT cross-sell and up-sell project as **`xsus_subsrvc_project`**.

Create a Subscription-Service Level Cross-Sell and Up-Sell Model in SAS Enterprise Miner

Log on to SAS Enterprise Miner

To create a cross-sell and up-sell model, log on to SAS Enterprise Miner with a profile and connect to a server.

Create a Project

1. Click **New Project**.
2. In the Create New Project wizard, complete these steps:
   a. Select the **SAS Server** and click **Next**.
   b. In the **Project Name** box, type *Cross-Sell and Up-Sell model for Prepaid Subscriptions*.
   c. In the **SAS Server Directory** box, enter the path for the **`model`** folder that is created in the DABT project workspace. For example, you can type the path as **`<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_subsrvc_project/model`**.
   d. Click **Next**.
   e. Do not change the default folder location that is displayed for **SAS Folder Location** and click **Next**.
   f. Click **Finish**. This step creates the SAS Enterprise Miner project and also the project workspace.
3. Select the *Cross-Sell and Up-Sell Model for Prepaid Subscriptions* project and then in the **Property** sheet, click **Project Start Code**.
4. In the Project Start Code window, copy the code from the cfdn_xsus_em_startup.sas file. This code is available in the **`<SAS configuration path>/Lev1/`**

`SASApp/SASEnvironment/SASOfferOptForCommunication5.2/SASCode/insertscripts` folder.

Note: Make sure that you enter proper values for all the parameters in the start-up code. Whenever the SAS Enterprise Miner model details are being written back for the first time, the value of cfdn_reexec_flg should be N. For all the subsequent runs, the value of cfdn_reexec_flg should be Y.

5. Click **Run Now**.
6. Click **Log** to view the log file and make sure that the log file does not contain any errors.
7. Click **OK**.

Create Diagram

1. In the project tree, right-click **Diagrams** and select **Import Diagram from xml**.
2. Select the **`xsus_subsrvc_diag.xml`** that you created when you imported the sample model.
3. Click **Open**. The diagram is imported and opened in the Diagram pane.

Create Data Source

1. Right-click **Data Source** and in the Property window, select **Create Data Source**.
2. In the Data Source Wizard, complete these steps:
   a. Select the default value (SAS Table) for the metadata source and click **Next**.
   b. Click **Browse** to select a SAS table. All the predefined SAS libraries are displayed.
   c. Double-click the library in which you stored the modeling ABT for subscription-service level cross-sell and up-sell and select the appropriate data set. For details, see "Construct ABT" on page 139. For example, the data set can be xsus_subsrvc_data. The subsequent steps refer to this data set.
   d. Click **OK**.
   e. Click **Next**. The table properties window appears.
   f. Click **Next**.
   g. Select **`Advanced`** from the metadata advisor options and click **Next**. The details of variables included in the ABT are displayed.
   h. Set the roles of the variables as explained below.

   Note: The following table indicates the roles of only the important variables. For the other variables, the role can be ID or Input. Make sure that the level of the target variable is binary.

Table 8.4 *Variables and Roles*

Variable Name	Role
XUP_SRVC_FLG	Target
AVG_REV_AMT	Input

Note: Services are activated at subscription level. Therefore, the cross-sell and up-sell scores for services are computed at subscription level. However, it needs to be aggregated at customer level. The usage revenue amount at subscription level is represented in the AVG_REV_AMT column. There can be a scenario in which a customer has multiple subscriptions. In this case, the proportion of AVG_REV_AMT at subscription level is considered to provide weights for cross-sell and up-sell scores, which need to be aggregated at customer level. Therefore, the AVG_REV_AMT column is mandatory. The column name might be different in the Analytics data mart. You need to provide the correct column name that indicates the average revenue for every subscription. This column should be populated as a part of the ABT building activity. Also, you can configure the name of this column. In the start-up code of the SAS Enterprise Miner project for which you have built the model, you have to provide the name of this column. The column name must be same as the value of the cfdn_avg_rev_var_nm macro variable.

i. In the Decision Configuration page, select **No**, and then click **Next**.

j. Click **Next**. The information about the ABT variables is displayed.

k. Click **Finish**.

Build Subscription-Service-Level Cross-Sell and Up-Sell Model

1. Delete the input data set of the diagram that was created in the Create Diagram step.
2. Drag and drop the xsus_subsrvc_data data set in the **Diagram** pane and connect it to the first node of the diagram.
3. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Train`**.
4. Delete the input data set of the diagram that is connected to the score node.
5. Drag and drop the xsus_subsrvc_data data set in the **Diagram** pane and connect it to the score node of the diagram.
6. In the **Train** section of the Property and Value table, make sure that the role of the data set is set to **`Score`**.
7. Right-click the xsus_subsrvc_data data set, which is connected to score node and click **Edit Variables**.
8. Set the role of the XUP_SRVC_FLG variable to **`Rejected`**.
9. Right-click the **Model Writeback** node and select **Update**.
10. After the diagram is updated successfully, right-click the **Model Writeback** node and select **Run**. Select **Yes** when asked for confirmation.
11. Click **OK** when the run is complete. In this step, the following tasks are completed:
    - The ANALYTICAL_MODEL_D table of the Foundation data mart and ANALYTICAL_MODEL_MASTER table of the Analytics data mart are updated.
    - The association between the model and the service for which the model is being built is stored in the ANALYTICAL_MODEL_DTL table of the Foundation data mart.
    - The optimizedcode.sas score code is generated in the **`<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_subsrvc_project/scrcd`** folder.

- The significant variables of the model are written in the abt_sgnfcnt_vars data set. This data set is available in the following location: **<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_subsrvc_project/scrcd**
- The xsus_subsrvc_project_scr scoring project is registered for the xsus_subsrvc_data modeling ABT in the PROJECT_MASTER table of the Analytics data mart.
- In the batch run, the xsus_subsrvc_data_scr scoring ABT is generated in the following location: **<SAS configuration path>/Applications/SASOfferOptforComm5.2/Data/common_data/analytics/xu/xsus_subsrvc_project/abt**.
- The significant variables are added in the ABT_X_VARIABLE table of the Analytics data mart against the scoring ABT (xsus_subsrvc_data_scr).

Batch Run for Cross-Sell and Up-Sell

Procedure

After the ABT for cross-sell and up-sell is built, the analytical model is built in SAS Enterprise Miner. This model can be deployed for generating scores for the subsequent runs. In order to complete this task, you have to run the batch macro cfdn_batch_macro.sas, which is called in the cfdn_outer_macro.

This section details the steps that you must follow for the scores writeback (also called batch run) for the cross-sell and up-sell module.

1. Go to the**<SAS configuration path>/Lev1/<SAS Application Server context name>**.
2. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file, respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
3. Open the cfdn_outer_macro.sas file, which is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts** folder.
4. Associate the subset query with the scoring project.

   Note: Complete the instructions for this step if you want to apply a different subset criterion (other than what you have defined for modeling ABT) to the scoring ABT.

   a. Find the call for the cfdn_update_project_subset_query macro in the cfdn_outer_macro. If you are unable to locate this call, then add it in the cfdn_outer_macro macro.

   b. Assign the name of the scoring project to the cfdn_project_nm the macro argument.

   c. Assign the name of the subsetting query that you want to associate with the scoring project to the cfdn_query_nm macro argument.

d. Run the cfdn_update_project_subset_query macro. View the logs through Base SAS.

e. Verify that the appropriate subset query is associated with the scoring project in the DABT_ADM.PROJECT_MASTER table.

5. Execute the batch run for the project.

a. The cfdn_batch_macro macro executes the scoring run of a DABT project. In order to run this macro, specify appropriate values for the following arguments:

project_sk
surrogate key of the project that is to be executed in the batch run.

abt_sk
surrogate key of the corresponding scoring ABT.

cfdn_avg_rev_var_nm
variable name (corresponding to variable_column_nm of the DABT_ADM.VARIABLE_MASTER table) for the average revenue column, which will be used while aggregating the scores from subscription level to customer level. This aggregation is applicable to cross-sell and up-sell at subscription-service level and customer-offer level. Therefore, a correct value must be passed for this argument irrespective of whether the cross-sell and up-sell is at customer-offer or subscription-service level. Also, the column name that appears here must be a part of the scoring ABT represented by abt_sk provided.

execution_dt
the supplied value for this argument indicates the date as of which the ABT needs to be built. If a value is not supplied, then the ABT will be built with the current date.

b. In Base SAS, open a new program.

c. Type the following lines of code in the program:

```
option mlogic mprint symbolgen;
%cfdn_batch_macro(project_sk=17,abt_sk=7,cfdn_avg_rev_var_nm=AVG_PSU_URVAT_L6M);
```

d. Save the file at a suitable location.

e. Go to **<SAS configuration path>/Lev1/SASApp/BatchServer** and invoke the sasbatch.bat or sasbatch.sh file.

f. Invoke the SAS code that you saved in step d.

g. View the log file that is created in the following folder: **<SAS configuration path>/Lev1/Applications/SASOfferOptForCommServer5.2/Logs/dabt_logs/flex/today's date**. The log file will be created with the name **analytics_batch_<scoring_project_sk><scoring_abt_sk>**. Here, the **<scoring_project_sk>** is the surrogate key of the project that you scheduled, whereas **<scoring_abt_sk>** is the surrogate key of the corresponding scoring ABT.

Details

In the batch run, the following steps are executed:

1. Based on the scoring ABT sk and the execution date that are provided as the macro arguments to the cfdn_batch_run macro, the build_abt macro is called. As a result,

the scoring ABT is generated in the **abt** folder of the corresponding DABT project workspace

2. After the scoring ABT is generated, the corresponding optimized score code is applied. There is always only one score code available for a given scoring ABT.
3. After the scored ABT is created, the scores are written back to the Foundation data mart. The cfdn_batch_macro searches for the purpose of the project under which the modeling ABT is created. If the purpose is for a cross-sell and up-sell module, then scores and other information is written back to the following tables of the Foundation data mart.

   For the customer-offer level model, scores are written back to the CUST_MODEL_STEPLEVEL_SCORE_DTL table. The information of the scoring run such as the analytical model sk and the score date is written back to PREDICTIVE_MODEL_RUN_DTL every time the model is scored.

   For the subscription-service level mode, scores are written back to the SUBSCRP_SCORE_STEPLEVEL_DTL table. If the scores aggregation flag is set to Y, then these scores are aggregated to customer level. The aggregated scores are then written back to the CUST_MODEL_STEPLEVEL_SCORE_DTL table of the Foundation data mart. The information of the scoring run such as the analytical model sk and the score date is written back to the PREDICTIVE_MODEL_RUN_DTL table, every time the model is scored.

Chapter 9

Solution-Specific ETL Jobs

ETL Jobs for Populating Application Data Mart

SAS Offer Optimization for Communications requires certain data for generating workflow reports and running application processes. This activity is completed by running a set of jobs. These jobs populate data from the Foundation data mart into the Application data mart. In addition, these jobs write back the data that describes business groups and the association between business groups and customers to the Foundation data mart tables.

Table 9.1 *ETL Jobs for Application Data Mart*

Job Name	Purpose	Primary Source Table	Target Table
bpp_cust_dtl_job	gets the required customer-related information in the application area so that further application process can access all the data from this location instead of the Foundation data mart. This job should be synchronized with the loading frequency of the Foundation data mart.	CUST_D	CUST_DTL
bpp_bg_write_back_job	writes back the metadata of the business group that is created in the application. You must run this job when a new business group is created or an existing business group is modified.	BSNSGRP_SUMMARY	BUSINESS_GROUP_D
bpp_cust_bg_association_job	writes back the association of the business group with the customer in the Foundation data mart tables.	BSNSGRP_FLTR_SUMMARY,CUST_D	CUST_X_BUSINESS_GROUP_BRIDGE

Job Name	Purpose	Primary Source Table	Target Table
bpp_cust_rf_key_tmp_mth_job	loads the latest (based on time grain) keys and dimensional values in a temporary table in order to ensure processing efficiency.	CUST_MTH_SUMMARY_F	CUST_RF_KEY_TMP
bpp_cust_rf_key_tmp_weekly_job	loads the CUST_SERVICE_MONTHLY_RF table, which is used by the process that creates application reports.	CUST_WEEKLY_SUMMARY_F	CUST_RF_KEY_TMP
bpp_cust_srvc_mthly_summary_rf_job	loads the latest (based on time grain) keys and dimensional values in a temporary table in order to ensure processing efficiency.	CUST_SERVICE_MTH_F	CUST_SERVICE_MONTHLY_RF
bpp_cust_srvc_rf_key_tmp_mth_job	loads the latest (based on time grain) keys and dimensional values in a temporary table in order to ensure processing efficiency.	CUST_SERVICE_MTH_F	CUST_RF_KEY_TMP
bpp_cust_srvc_rf_key_tmp_wkly_job	loads the latest (based on time grain) keys and dimensional values in a temporary table in order to ensure processing efficiency.	CUST_SERVICE_WEEKLY_F	CUST_RF_KEY_TMP
bpp_cust_srvc_wk_summary_rf_job	loads the CUST_SERVICE_WEEKLY_RF table, which is used by the process to create application reports.	CUST_SERVICE_WEEKLY_F	CUST_SERVICE_WEEKLY_RF
bpp_cust_summary_rf_tmp_job	loads the latest keys and measure values in the temporary table in order to ensure processing efficiency.	CUST_RF_KEY_TMP,CUST_D	CUST_SUMMARY_RF_TMP
bpp_cust_weekly_summary_rf_job	loads the CUST_WEEKLY_SUMMARY_RF table, which will be use by the process that creates application reports.	CUST_WEEKLY_SUMMARY_F,CUST_SUMMARY_RF_TMP	CUST_WEEKLY_SUMMARY_RF

Job Name	Purpose	Primary Source Table	Target Table
bpp_cust_mthly_summary_rf_job	loads the CUST_MONTHLY_SUMMARY_RF table, which will be used by the process that creates application reports.	CUST_MTH_SUMMARY_F,CUST_SUMMARY_RF_TMP	CUST_MONTHLY_SUMMARY_RF

Chapter 10

Performing Middle-Tier Administrative Tasks

Middle-Tier Administration Overview

The middle-tier component synchronizes all components of SAS Offer Optimization for Communications and enables them to function together as an integrated system. It interacts with the Web-based user interface of SAS Offer Optimization for Communications and the database server. Also, it connects to the SAS run-time environment in order to run analytical processes.

You have to perform certain administrative activities in order to set up and configure the middle-tier component on various servers such as the metadata server, the application server, and the data server. In addition, you have to maintain the configuration file that enables you to configure the middle-tier component and also the error logs that are generated by SAS Offer Optimization for Communications.

Configuring the BPPConfiguration File

The BPPConfiguration file contains information about the properties that you can configure for SAS Offer Optimization. You can configure these properties according to your requirements. However, it is recommended that you should not change the default values that are set for all properties. Any uninformed change can result in fatal errors and loss of data integrity.

The default location of the BPPConfiguration file is the WebApp.ConfigHome software component property. You can change this location according to your requirements.

The BPPConfiguration file is divided into certain property groups. Each property group serves a specific purpose and enables you to configure a set of properties.

Note: The table below does not explain all the property groups that are defined in the BPPConfiguration file. Therefore, make sure that you do not modify the properties that are not listed below.

Table 10.1 *BPPConfiguration File*

Property Group Name	Purpose
BPP Environment	configures the name of the application.
Data sources	refers to the JNDI name of the data source. You must not change the properties of this group.
Status Codes	externalized configurations that are used internally by the application. You must not change the properties of this group.
BPP User Roles	indicates the role names that are used to identify users of the application based on the predefined roles—administrators, analysts, and business users.
BPP Functional	includes properties that are internal to the application. Therefore, except for the BPP_CACHE_INTERVAL property, do not change any of the properties of this group. The BPP_CACHE_INTERVAL property is configured in milliseconds. This property indicates the cache interval. Some master tables that are frequently used are cached by the middle-tier. The default cache interval is one day; after that, the cache is refreshed. You can increase or decrease the value for the cache interval.
BPP Miscellaneous	includes properties that are not currently used.

Property Group Name	Purpose
FlexConfiguration	contains the following properties: BPP_THEMES_ENABLED loads the Flex themes. Do not change the default value for this property. BPP_THEMES_DEFAULT specifies the default Flex theme. BPP_TIMEOUT_ENABLED enables you to define session time-out settings. BPP_TIMEOUT_WARNING_DURATION the duration of time in seconds for displaying the warning dialog box before the time-out. BPP_TIMEOUT_PINGWINDOW_INTERVAL the time in milliseconds for keeping the Flex session and server session alive for any user action that is taken on the Flex side. BPP_SESSION_TOUCH_ENABLED enables you to configure the session touch action for synchronizing the client and server sessions. BPP_LOCALIZATION_ENABLED enables you to support the localization feature. BPP_LOCALE specifies the fallback locale, which is used if no locale is defined in the request or the locale that is defined in the request is not supported. BPP_SERVER_PROTOCOL used to create dynamic channel set for creating a remote connection from the application interface to the server. Do not change the default value for this property. BPP_SERVER_PORT used to create a dynamic channel set for creating a remote connection from the application interface to a server. Do not change the default value for this property. BPP_SERVER_NAME used to create a dynamic channel set for creating a remote connection from the application interface to a server. Do not change the default value for this property. BPP_SERVER_CONTEXT used to create a dynamic channel set for creating a remote connection from the application interface to a server. Do not change the default value for this property. BPP_CAPABILITIES_ENABLED indicates whether the capabilities are used in order to regulate user actions. Do not change the default value for this property. BPP_HELP_ENABLED enables you to define a help-related configuration. BPP_SUPPORT_URL indicates the URL that is to be used for the target of the link in the Help menu. BPP_ADMIN_EMAIL_ID specifies the e-mail ID that is used for the send e-mail notification feature that is available in the SAS Offer Optimization for Communications interface.

Property Group Name	Purpose
BPPStoredProcedures	used by the application middle-tier in order to execute the stored procedures. Each property of this group configures a single stored procedure that is initiated by the middle-tier for a specific functional goal.
HibernateConfiguration:	SAS Offer Optimization for Communications uses the Hibernate object relational mapping tool for accessing the database. This group includes properties that Hibernate uses in order to connect, optimize, and map with the table and log details of the execution. You can change only the following properties of this group: • hibernate.show_sql • hibernate.format_sql • hibernate.use_sql_comments These properties can be used collectively to provide execution details of database queries.

Middle-Tier and Database Connectivity

SAS Offer Optimization for Communications will be deployed on the application server. SAS Offer Optimization for Communications uses JNDI lookup to connect to the database. The application searches for the JNDI name BPPDS to identify the datasource that it needs to connect to.

The middle-tier of SAS Offer Optimization for Communications is connected to the database using the application-server-managed datasource. For this purpose, you have to create the datasource of the SAS Offer Optimization for Communications database in the application server and configure its JNDI lookup as BPPDS. Therefore, set the JNDI name of the datasource as *BPPDS*.

Here are the application-server-specific details of managing the datasource:

JBOSS
: On a Windows machine, locate the BPPDataSource-ds.xml file in the **`<JBOSS_HOME>/server/SASServer11/deploy`** folder. For example, this location can be **`C:/jboss-4.2.0.GA/server/SASServer11/deploy`**.

WebLogic and WebSphere
: Go to the application server administration console. Complete the required configuration for the JNDI resources.

Middle-Tier on Metadata Server

Updating the WebApp.ConfigHome Property

The BPPConfiguration file is stored in a default location. If you move this file to another location, you must change the WebApp.ConfigHome property of the software component in SAS Management Console.

To update the WebApp.ConfigHome property:

1. Log on to SAS Management Console and connect to a profile.
2. On the **Plug-ins** tab, expand **Application Management** ⇨ **Configuration Manager**.
3. Locate the software component of SAS Offer Optimization for Communications, which is named **Offer Opt For Comm 5.2**.
4. Right-click **Offer Opt For Comm 5.2** and select **Properties**.
5. On the **Advanced** tab, locate the WebApp.ConfigHome property. This is the only application-specific property that is defined for this software component.
6. Make sure that the location of the BPPConfiguration file is correct. Make appropriate changes if required.

Roles and Capabilities

SAS Offer Optimization for Communications has three predefined roles that can be assigned to users. Each role is assigned a set of predefined capabilities. One or more roles can be assigned to a user who can access SAS Offer Optimization for Communications. If multiple roles are assigned to a user, then the least restrictive capability of each role is granted to the user.

SAS Offer Optimization has the following predefined roles.

Offer Opt for Comm: Administration
: Users who are assigned this role can access and perform all functions that are performed by using the SAS Offer Optimization for Communications interface. They can view, create, edit, and delete business groups and projects and assign users to business groups. In addition, they can also view, create, edit, and delete workflow reports.

Offer Opt for Comm: Analysis
: Users who are assigned this role can access the projects that are assigned to them and perform workflow steps. They can view details of business groups that are assigned to them and projects that are shared by other users. In addition, they can also create, modify, and delete the workflow reports for the projects that are assigned to them.

Offer Opt for Comm: Viewing
: Users who are assigned this role can perform basic tasks such as view business groups, projects, and reports. Therefore, they cannot change the state of SAS Offer Optimization for Communications.

Creating Users and Granting Roles and Capabilities

You have to define users who can log on to SAS Offer Optimization for Communications and perform tasks based on the role that is assigned to them. For details about defining users and granting roles and capabilities, see *SAS 9.2 Management Console Guide to Users and Permissions*. This document is available at the following location: **http://support.sas.com/documentation/cdl/en/mcsecug/61708/PDF/default/mcsecug.pdf**.

Logging Configuration Administration

The logs of the middle-tier component are maintained in the **<SAS configuration path>/Lev1/Web/Logs** folder. For example, on a Windows machine, this location can be **C:/SAS/EBIEDIEG/Lev1/Web/Logs**.

The logs of the middle-tier component use log4j and therefore support the following features:

- Increase or decrease the level of detail that is generated in the logs.
- Define the location where the logs can be generated.
- Select the preferred format for the log files.

The log configuration file is maintained in the **<SAS configuration path>/Web/Common/LogConfig** folder. For example, this location can be **C:/SAS/EBIEDIEG/Lev1/Web/Common/LogConfig**. You can change the logging configuration for the middle-tier component. However, before you do so, refer to the log4j documentation. For details, see **http://logging.apache.org/log4j/**.

Middle-Tier Error Messages

The following table lists the codes and the descriptions of errors that can occur when the SAS Offer Optimization for Communications interface interacts with the middle-tier component. When an error occurs, the respective error code is present in the application-specific middle tier log. You can refer to the error descriptions that are listed below when you debug the error.

Table 10.2 *Error Messages*

Error Code	Description
ER0000	An error has occurred. Please contact the system administrator.
ER0001	Error while inserting data
ER0002	Error while inserting or updating data
ER0003	Error while generating ID

Error Code	Description
ER0004	Error while deleting data
ER0005	Error querying data
ER0006	Error loading data
ER0007	Error while updating data
ER0008	Error retrieving session from Flex Context
ER0009	Could not resolve user
ER0010	Error while initializing
ER0011	Configuration not found
ER0012	Error Reading Config file
ER0013	Not able to verify product license
ER0014	Stored process does not exist.
ER0015	Stored Process Name search yielded null
ER0016	Stored Process Name search yielded zero results
ER0017	Error Executing Stored procedure
ER0018	Error Destroying Execution2Interface
ER0019	Project Name already exists. Please select a different project name
ER0020	Project is currently in execution.
ER0021	Error retrieving user capabilities
ER0022	Error retrieving user roles
ER0023	No property found
ER0024	Business Group Already exists. please change the name to create a new one
ER0025	Not able to clone Expression object
ER0026	Product is not valid.
ER0027	Product is not licensed.
ER0028	Product license has expired.

Error Code	Description
ER0029	Display themes cannot be found. Please verify that the installation included Flex Application Themes.
ER0030	This business group has one or more projects in the batch mode. Please pull these projects back to the design mode before you delete this business group
ER0031	You are not authorized to create a project in this business group
ER0032	Unable to retrieve users.
ER0033	An internal error has occurred when you were trying to execute. Please contact the system administrator.

Troubleshooting

Problem

I am able to see the application. However, when I log on, I see some pop up-error messages, and then the application window appears. Also, I cannot see any business groups or projects in the respective sections of the navigation pane.

Solution

1. Make sure that the database has started and the application server is able to connect to the database.
2. Verify that the HibernateConfiguration is pointing to the correct JNDI name.
3. Enable ROOT logging to DEBUG in the middle-tier log configuration file that is maintained at the**`<SAS configuration path>\Web\Common\LogConfig`** location.

View the details of the errors that are generated in the log file.

Chapter 11

Business Groups Administration

Configuration for Business Groups

Business groups are defined and maintained by using the SAS Offer Optimization for Communications interface. However, in order to define business groups, you have to complete certain prerequisite activities.

In order to enable users to define business groups, you have to complete the following tasks:

- Register the stored procedures.
- Configure variables.
- Configure values for variables.

- Define filter combinations.

Note: You might have to consult your business analyst when you perform these tasks. Also, it is recommended that you be familiar with the structures of the main and reference tables of the Foundation data mart and the Application data mart tables.

Registering the Stored Procedure for Business Groups

Overview

SAS Offer Optimization for Communications requires the bpp_spexecutecreatebsnsgrpfltr.sas stored procedure for creating business groups. This stored procedure is available in the **`<SAS installation path>/SASFoundation/9.2/bppsrv/sasstp`** folder. Before you use this stored procedure, you have to register it in SAS Management Console.

Register the Stored Procedure

To register the stored procedure:

1. Log on to SAS Management Console and connect to a profile.
2. If you are registering the stored procedure for the first time, then create a new folder in **Products** ⇨ **Offer Optimization for Communications**.
3. Right-click the folder that you have created. On the **Actions** menu, select **New Stored Process**.
4. Specify the name and description of the stored procedure and click **Next**.
5. Select **`SAS server as SASApp - Logical Stored Process Server`**.
6. Select **Source code repository path** in order to point to the **`<SAS installation path>/SASFoundation/9.2/bppsrv/sasstp`** location.
7. Specify the name of the source file of the stored procedure that you want to register.
8. Click **Next**.
9. Click **New Prompt** if the stored procedure requires some initial parameters.
10. Click **Finish**.

Execute the Stored Procedure

You can execute the stored procedure that you have registered only after you configure variables and values for business groups. For details, see "Configuring Variables and Variable Values for Business Groups" on page 165.

To execute the stored procedure:

1. Open SAS Enterprise Guide.
2. Go to the folder in which you have registered the stored procedure. This folder is available in **Products** ⇨ **SAS Offer Optimization for Communications**.

3. Drag and drop the stored procedure into the Process flow diagram.
4. Right-click the stored procedure and select **Run** to execute it.
5. Specify appropriate values if you are prompted for certain parameters.

Configuring Variables and Variable Values for Business Groups

Overview

Each business group that is defined using the SAS Offer Optimization for Communications interface is associated with certain selection criteria. Based on these selection criteria, customers are added to the business group. For details, see *SAS Offer Optimization for Communications User's Guide*. In order to enable users to define the selection criteria, you have to configure certain variables and define their values.

Here is the list of Application data mart tables that you have to configure in order to define variables and values for business groups.

Table 11.1 *Tables for Business Groups Variables*

Table Name	Description
BPP_VRBL_MASTER	contains the master list of variables that are required for SAS Offer Optimization for Communications configuration. This table is populated with default data. Make sure that you do not delete the default data.
BPP_VRBL_MASTER_NLS	contains the localized names for variables that are defined in the BPP_VRBL_MASTER table.
BSNSGRP_VRBL_MASTER	contains the variables that you want to consider for defining business groups. This table is populated with default data. Make sure that you do not delete the default data.
BPP_REF_VRBL_VAL_MASTER	contains values of variables that you define in the BPP_VRBL_MASTER table. This table is populated with default data. Make sure that you do not delete the default data.
BPP_REF_VRBL_VAL_MASTER_NLS	contains the localized value names for values that are defined in the BPP_REF_VRBL_VAL_MASTER table.
BSNSGRP_REF_VRBL_VAL_MASTER	contains values of variables that you define in the BSNSGRP_VRBL_MASTER table. This table is populated with default data. Make sure that you do not delete the default data.

Assumptions

When you define a variable for business groups, make sure that a customer has only one value of this variable. For example, a customer can have multiple account types. Therefore, you must not select the account type variable for defining business groups.

In addition, for defining business groups, you must consider only those variables that are defined in the corresponding reference tables and main dimensions of the Foundation data mart.

For example, here is the list of variables that you can define for business groups based on these assumptions:

- Customer type
- Payment mode
- Customer geography
- Offer segment

Configure Variables

A default set of variables is defined in the BPP_VRBL_MASTER table. However, you can also decide on the variables that you want to consider for business groups. Define these variables in the BPP_VRBL_MASTER table. Make sure that you specify values for the following columns of the BPP_VRBL_MASTER table.

Table 11.2 *Required Columns of BPP_VRBL_MASTER Table*

Column	Description
VARIABLE_ID	the sequence number that uniquely identifies the variable.
VARIABLE_NM	the name of the variable. Make sure that you refer to the corresponding reference table of the Foundation data mart when you define this column value. The value in this column must be the same as the name of the column that contains the reference code of the reference table.
SRC_TABLE_NM	the name of the table that sources the association of the variable.
SRC_COLUMN_NAME	the name of the column of the source table.
SRC_LIBRARY_NM	the name of the source library that associates the variable with the application.
REF_SRC_TABLE_NM	the name of the reference table of the Foundation data mart from which the values for this variable are populated.

Column	Description
REF_SRC_TABLE_COLUMN	the name of the column of the reference table from which values for this variable are populated.
REF_LIBRARY_NAME	the name of the source library from which values for this variable are populated.

The following table lists the columns of the BPP_VRBL_MASTER table that you have to configure in order to define payment mode as one of the variables for business groups.

Table 11.3 *Sample Data in BPP_VRBL_MASTER Table*

Column	Value
VARIABLE_ID	1
VARIABLE_NM	PYMNT_MODE_CD
SRC_TABLE_NM	OFFER_BUNDLE_D
SRC_COLUMN_NAME	BASE_OFFER_PYMNT_MODE_CD
SRC_LIBRARY_NM	CFDDIM
REF_SRC_TABLE_NM	PAYMENT_MODE
REF_SRC_TABLE_COLUMN	PYMNT_MODE_CD
REF_LIBRARY_NAME	CFDREF

After you configure values in the BPP_VRBL_MASTER table, make sure that you enter correct values for these variables in the BPP_VRBL_MASTER_NLS table. To do so, in the BPP_VRBL_MASTER_NLS table, enter appropriate values for the following columns.

Table 11.4 *BPP_VRBL_MASTER_NLS Table*

Column	Description
VARIABLE_ID	the sequence number that uniquely identifies the variable.
LOCALE	a specific language code that is associated with a geographical region.
VRBL_DSPLY_NM	the localized display name that is associated with the variable.
VRBL_DSPLY_DESC	the localized display description that is associated with the variable.

Example: Configure BPP_VRBL_MASTER_NLS Table

To configure the payment mode variable, you can enter the following data in the BPP_VRBL_MASTER_NLS table.

Table 11.5 *Sample Data in BPP_VRBL_MASTER_NLS Table*

Column	Value
VARIABLE_ID	1
LOCALE	en_US
VRBL_DSPLY_NM	Payment mode
VRBL_DSPLY_DESC	Payment mode

After you configure variables in the BPP_VRBL_MASTER table, identify the variables that you want to consider for defining selection criteria for business groups. In the BSNSGRP_VRBL_MASTER table, enter the IDs of these variables. For example, for the payment mode variable, enter the VARIABLE_ID that you have entered in the BPP_VRBL_MASTER table into the VARIABLE_ID column of the BSNSGRP_VRBL_MASTER table.

Configure Values for Variables

After you define variables for business groups, configure values for these variables in the BPP_REF_VRBL_VAL_MASTER table. Make sure that you specify values for the following columns of the BPP_REF_VRBL_VAL_MASTER table.

Table 11.6 *Required Columns of BPP_REF_VRBL_VAL_MASTER Table*

Column	Description
VARIABLE_ID	the unique sequence number of the variable as defined in the BPP_VRBL_MASTER table.
BPP_REF_VARIABLE_VALUE_ID	the sequence number that uniquely identifies the value of a certain variable.
BPP_REF_VARIABLE_VALUE	the value of the variable. Make sure that you refer to the corresponding reference table of the Foundation data mart when you define this column value. The value in this column must be same as the column value of the corresponding reference table of the Foundation data mart.

Example: Configure BPP_REF_VRBL_VAL_MASTER Table

To configure values for the payment mode variable, you can enter the following data in the BPP_REF_VRBL_VAL_MASTER table.

Table 11.7 *Sample Data in BPP_REF_VRBL_VAL_MASTER Table*

Column	VARIABLE_ID	BPP_REF_VARIABLE_VALUE_ID	BPP_REF_VARIABLE_VALUE
Record 1	1	1	PREPAID
Record 2	1	2	POSTPAID

After you configure values in the BPP_REF_VRBL_VAL_MASTER table, make sure that you enter correct values for the configured values in BPP_REF_VRBL_VAL_MASTER_NLS table. To do so, in the BPP_REF_VRBL_VAL_MASTER_NLS table, enter appropriate values for the following columns.

Table 11.8 *Sample Data BPP_REF_VRBL_VAL_MASTER_NLS in Table*

Column	Description
BPP_REF_VARIABLE_VALUE_ID	the unique sequence number of the variable value as defined in the BPP_REF_VRBL_VAL_MASTER table.
LOCALE	a specific language code that is associated with a geographical region.
BPP_REF_VRBL_VAL_DSPLY_NM	the localized display name that is associated with the variable value.
BPP_REF_VRBL_VAL_DSPLY_DESC	the localized display description that is associated with the variable value.

Example: Configure BPP_REF_VRBL_VAL_MASTER_NLS Table

After configuring values for the payment mode variable, you can enter the following data in the BPP_REF_VRBL_VAL_MASTER_NLS table.

Table 11.9 *Sample Data in BPP_REF_VRBL_VAL_MASTER_NLS Table*

Column	BPP_REF_VARIABLE_VALUE_ID	LOCALE	BPP_REF_VRBL_VAL_DSPLY_NM	BPP_REF_VRBL_VAL_DSPLY_DESC
Record 1	1	en_US	Prepaid	Prepaid
Record 2	2	en_US	Postpaid	Postpaid

After you configure values in the BPP_REF_VRBL_VAL_MASTER table, identify the values that you want to consider for defining selection criteria for business groups. To do so, in the BSNSGRP_REF_VRBL_VAL_MASTER table, enter appropriate values for the following columns.

Table 11.10 Columns of BSNSGRP_REF_VRBL_VAL_MASTER Table

Column	Description
BPP_REF_VARIABLE_VALUE_ID	the sequence number that uniquely identifies the value of the variable as defined in the BPP_REF_VRBL_VAL_MASTER table.
VARIABLE_ID	the unique sequence number of the variable as defined in the BPP_REF_VRBL_VAL_MASTER table.
VARIABLE_VALUE_DLTD_IND	indicates whether the variable value is to be considered for defining selection criteria of business groups. If you want to consider this value for defining the selection criteria, set the value for this column to NULL.

Example: Configure BSNSGRP_REF_VRBL_VAL_MASTER Table

After configuring values for the payment mode variable, you can enter the following values in the BSNSGRP_REF_VRBL_VAL_MASTER table.

Table 11.11 Sample Data in BSNSGRP_REF_VRBL_VAL_MASTER Table

Column	Value
VARIABLE_ID	1
BPP_REF_VARIABLE_VALUE_ID	1
VARIABLE_ID	1
BPP_REF_VARIABLE_VALUE_ID	2

Constraints

You have to decide on the maximum number of variables and values that you can configure in the BSNSGRP_VRBL_MASTER and BSNSGRP_REF_VRBL_VAL_MASTER tables respectively. This consideration has an impact on the exploration of the hierarchical list. Therefore, it is recommended that you restrict the number of variables and values to four or five.

Defining Filter Combinations for a Business Group

After you define variables and their values in the BSNSGRP_VRBL_MASTER and BSNSGRP_REF_VRBL_VAL_MASTER tables, you have to run the bpp_spexecutecreatebsnsgrpfltr.sas stored procedure to set up all possible filter

combinations. These filter combinations are then stored in BSNSGRP_FLTR_SUMMARY and BSNSGRP_FLTR_X_VRBL_VAL tables.

To define filter combinations for a business group:

1. Make sure that you have populated the following tables with correct data:
   - BSNSGRP_VRBL_MASTER
   - BSNSGRP_REF_VRBL_VAL_MASTER
2. Run the bpp_spexecutecreatebsnsgrpfltr.sas stored procedure, which is located in the **<SAS installation path>/SASFoundation/9.2/bppsrv/sasstp** folder. This step populates BSNSGRP_FLTR_SUMMARY and BSNSGRP_FLTR_X_VRBL_VAL tables with all combinations of variables and values that can be used as filter combinations.
3. In the BSNSGRP_FLTR_SUMMARY table, identify the filter combinations that you do not want to add as selection criteria for creating business groups. These combinations might not be significant for defining a business group. For example, a certain offer segment might not be applicable in certain geographical areas.
4. For each combination that you have identified in the above step, set the value of the BSNSGRP_FLTR_VALID_IND column to N.

   Note: If you set up this value after you define one or more business groups, then this change will not be applicable to the business groups that you have already defined. Therefore, make sure that you set up the value for the BSNSGRP_FLTR_VALID_IND column before you define business groups.

Defining Business Groups

After you complete the prerequisite configuration tasks, you can define business groups by using the SAS Offer Optimization for Communications interface. The variables and values that you have configured in the BSNSGRP_VRBL_MASTER and BSNSGRP_REF_VRBL_VAL_MASTER tables will be available in the hierarchical lists that you use to define the selection criteria. For details, see *SAS Offer Optimization for Communications: User's Guide.*

Editing a Business Group Scheme

Overview

The following tasks can necessitate changing a business group scheme:

- Add a variable to the BSNSGRP_VRBL_MASTER table.
- Delete a variable from the BSNSGRP_VRBL_MASTER table.
- Add a value to the BSNSGRP_REF_VRBL_VAL_MASTER table.
- Delete a value from the BSNSGRP_REF_VRBL_VAL_MASTER table.

These tasks impact the underlying business group scheme. Therefore, you must be very cautious when you perform these tasks. Also, before you perform these tasks, make sure that you archive all your project data.

Note: You have to define the filter combinations again after you perform any of these tasks. For details, see "Defining Filter Combinations for a Business Group" on page 170.

Adding or Deleting a Variable

A business requirement can necessitate adding or deleting variables that are defined for business groups. For example, in addition to the variables, payment mode, customer type, and geography, your business might want to consider a new variable such as an offer segment. In this case, you have to add a new variable to the BSNSGRP_VRBL_MASTER table. If you add or delete a variable, then you have to define all the business groups and projects again. Therefore, before you add a new variable to the BSNSGRP_VRBL_MASTER table or delete a variable from this table, make sure that you archive all your project data.

Adding a New Value for a Variable

A business requirement can necessitate adding one or more new values for a variable. For example, new states or regions can be added for the geography variable. For such business requirements, you have to add the new values in the BSNSGRP_REF_VRBL_VAL_MASTER table. When you add a new value for a variable, it is indicated in the hierarchical list of selection criteria that are defined by using theSAS Offer Optimization for Communications interface. You have to perform certain tasks such as run the business groups and the associated projects again. For details, see *SAS Offer Optimization for Communications: User's Guide*.

Deleting a Value for a Variable

A business requirement can necessitate removing one or more values for a variable. For example, certain states or regions might not be needed for the geography variable. For such business requirements, you have to update the BSNSGRP_REF_VRBL_VAL_MASTER table. To do so, set the VARIABLE_VALUE_DLTD_IND column to Y for the value that you do not want to consider for the selection criteria of business groups. As a result, the selection criteria that are defined for this value are deleted from the BSNSGRP_FLTR_X_VRBL_VAL , BSNSGRP_X_BPP_FLTR_X_CUST, and BSNSGRP_FLTR_SUMMARY tables. Therefore, you have to run the business groups and the associated projects again. For details, see *SAS Offer Optimization for Communications: User's Guide*.

Performing Tasks for a Deleted Business Group

Users can delete a business group using the SAS Offer Optimization for Communications interface. However, you have to perform certain administrative tasks before and after a business group is deleted.

If a business group is deleted, then the business group and also its projects are deleted. Therefore, before a business group is deleted, make sure that you archive the project data associated with the business group. After a business group is deleted, the selection criteria that are defined for the business group are assigned to the default business group. Therefore, the variable values that were selected for this business group are available for selection when the selection criteria for other business groups is defined.

Note: If you want to retrieve a business group that is deleted, in the BSNSGRP_SUMMARY table, change the value of the BSNSGRP_ACTIVE_IND column from *N* to *Y*. You have to retrieve the data of the projects that are associated with this business group. For details, see "Defining Filter Combinations for a Business Group" on page 170.

Viewing Log Files for Business Groups

When you run the stored procedure to define filter combinations or when you perform any task for a business group through the SAS Offer Optimization for Communications interface, a log file is created in the **<SAS configuration path>/Lev1/Applications/SASOfferOptForCommServer5.2/Logs/ooc_logs/bsnsgrplogs** folder. This log file contains information about all the tasks and the errors that might occur when you perform these tasks. For example, if you run the stored procedure to define the filter combinations, the bpp_spexecutecreatebsnsgrpfltr.log file is generated. Also, if you run a business group (with ID as 2) through the SAS Offer Optimization for Communications interface, the bpp_spexecutebsnsgrp2.log file is generated. Similarly, log files are generated when you perform other tasks through the SAS Offer Optimization for Communications interface.

Updating the Business Groups Data

Every time data is loaded into the Foundation data mart, you have to synchronize the business group tables with the latest data. This data synchronization is required to get the latest information about the customer base and offers.

To synchronize the business group tables with the Foundation mart tables:

1. Run the bpp_cust_dtl_job job. You can schedule this job to run after every load of the Foundation data mart.
2. Run the bpp_executeooc_bg_batch.sas macro. This macro is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASMacro** folder. You can schedule this macro or run it in the SAS Enterprise Guide environment.
3. Run the bpp_cust_bg_association_job job. You can schedule this job to run after the bpp_executeooc_bg_batch.sas macro is run successfully.

Chapter 12

Configuration for Application Workflow

Configuration for Workflow Steps

You have to perform certain configuration tasks before users create projects and work with the project workflow by using the SAS Offer Optimization for Communications interface. These configuration tasks are required for the target segment selection, microsegmentation, and offer ranking workflow steps.

Note: Users must not create projects and work with project workflow before the configuration tasks that are detailed in this chapter are complete. Otherwise, the workflow steps will not run successfully.

Configuration for Target Segment Selection Workflow Step

Variables Selection

Target segments are defined based on certain filter criteria. In order to define the filter criteria, you have to configure certain variables and define values for these variables. These variables are either of the categorical type or the band type. The variables that you define must be columns of the CUST_DTL table of the Application data mart.

Here is the list of columns of the CUST_DTL table from which you can select the variables for the target segment selection workflow step:

- CUST_STATUS_CD
- TENURE_ON_NETWORK_BAND
- GROSS_USG_ARPU_BAND
- CHURN_BAND_CD
- BILL_ARPU_BAND
- IND_CUST_GENDER_CD
- STATE_CD
- AGE_BAND_CD
- IND_CUST_MARITAL_STATUS_CD
- CITY_CD
- CUST_CREDIT_CLASS_CD
- IND_CUST_SEGMENT_TYPE_CD
- IND_CUST_STD_OCCUPATION_CD
- IND_OCCUPATION_INDUSTRY_GRP_CD
- IND_CUST_ETHINICITY_CD
- IND_CUST_EDUCATION_LEVEL_CD
- IND_CUST_INCOME_RANGE_CD

- ORG_CUST_ORG_TYPE_CD
- ORG_CUST_SEGMENT_TYPE_CD
- IND_CUST_CSP_EMPLOYEE_IND
- ACQUIRING_CAMPAIGN_CD
- BASE_OFFER_TYPE_CD
- BASE_OFFER_CUST_TYPE_CD
- BASE_OFFER_PYMNT_MODE_CD
- BASE_OFFER_SEGMENT_CD
- BPP_AVAILED_IND
- CUST_ADDRESS_POSTAL_CD
- COUNTY_CD
- REGION_CD
- COUNTRY_CD
- ANALYTICAL_MODEL_SEGMENT_CNCT
- CAMPAIGN_CD
- PROFITABILITY_BAND_CD
- DATA_USG_ARPU_BAND
- VOICE_USG_ARPU_BAND
- MESSAGE_USG_ARPU_BAND

Configure Variables

To configure variables for the target segment selection workflow step, complete the following steps:

1. Identify the columns of the CUST_DTL table that you want to consider as variables for the target segment selection workflow step. You might have to consult your business analyst to confirm the selection of variables.
2. In the BPP_VRBL_MASTER table, enter the details about the variables that you want to consider for the target segment selection workflow step.

You have to enter correct values for the following columns of the BPP_VRBL_MASTER table.

Table 12.1 *Columns of BPP_VRBL_MASTER*

Column	Description
VARIABLE_ID	the sequence number that uniquely identifies the variable.
VARIABLE_CD	the code that is assigned to the variable. This column is not used currently.

Column	Description
VARIABLE_NM	the name of the variable. Make sure that the value of this column is the same as the column name of the CUST_DTL table. Moreover, this has to be a categorical variable.
VARIABLE_PRFLNG_TYPE_CD	the profiling type code. Specify the value **CAT** to indicate that it is a categorical variable.
ANALYTICAL_VARIABLE_IND	indicates whether the variable is an analytical variable. Type **Y** to indicate that the variable is an analytical variable.
PREPAID_IND	indicates whether the variable can be used for target segments that contain prepaid customers. Type **Y** to indicate that the variable can be used for prepaid customers.
POSTPAID_IND	indicates whether the variable can be used for target segments that contain postpaid customers. Type **Y** to indicate that the variable can be used for postpaid customers.

In addition, the BPP_VRBL_MASTER table, contains the following columns:

- VARIABLE_DISPLAY_IND
- VARIABLE_DISPLAY_FORMAT
- VARIABLE_DATA_TYPE
- SRC_TABLE_NM
- SRC_COLUMN_NAME
- SRC_LIBRARY_NM
- REF_SRC_TABLE_NM
- REF_SRC_TABLE_COLUMN
- REF_LIBRARY_NAME
- RPT_SRC_TABLE_NM
- RPT_SRC_COLUMN_NM
- RPT_SRC_LEVEL
- VARIABLE_SUB_CTGRY_CD
- RPT_VRBL_IND
- CREATED_DTTM
- UPDATED_DTTM

However, you do not have to configure these columns.

Example: Configure BPP_VRBL_MASTER Table

You can configure the churn band variable by specifying the following values in the BPP_VRBL_MASTER table:

Table 12.2 *Configuration of Churn Band Variable*

Column	Value
VARIABLE_ID	15
VARIABLE_CD	NA
VARIABLE_NM	CHURN_BAND_CD
VARIABLE_PRFLNG_TYPE_CD	CAT
ANALYTICAL_VARIABLE_IND	N
PREPAID_IND	Y
POSTPAID_IND	Y

Note: You must also configure the localized values in the BPP_VRBL_MASTER_NLS table. For details, see "Configure Variables" on page 166.

Configure Values

After you configure the variables, you have to specify the values for each variable. To do so, enter appropriate details in the BPP_REF_VRBL_VAL_MASTER table.

Table 12.3 *Columns of BPP_REF_VRBL_VAL_MASTER Table*

Column	Description
VARIABLE_ID	the unique sequence number of the variable as specified in the BPP_VRBL_MASTER table.
VARIABLE_CD	the code that is assigned to the variable. This column is not used currently.
BPP_REF_VARIABLE_VALUE_ID	the sequence number that uniquely identifies the value.
BPP_REF_VARIABLE_VALUE	the reference value of the variable.

In addition, the BPP_REF_VRBL_VAL_MASTER table contains the following columns:

- BPP_REF_VARIABLE_VALUE_NM
- BPP_REF_VARIABLE_VALUE_DESC
- BPP_REF_VARIABLE_DISPLAY_IND

However, you do not have to configure these columns.

Example: Configure BPP_REF_VRBL_VAL_MASTER Table

You can configure the following reference values for the CHURN_BAND_CD variable.

Table 12.4 *Sample Data in BPP_REF_VRBL_VAL_MASTER Table*

Column	VARIABLE_ID	BPP_REF_VARIABLE_VALUE_ID	BPP_REF_VARIABLE_VALUE
Record 1	14	31	HIGH
Record 2	14	32	MED
Record 3	14	33	LOW

Note: You must also configure the localized values in the BPP_REF_VRBL_VAL_MASTER_NLS table. For details, see "Configure Values for Variables" on page 168.

Configuration for Microsegmentation and Offer Ranking Workflow Steps

Overview

Microsegmentation and offer ranking are the analytical components of SAS Offer Optimization for Communications that are integrated in the workflow.

You have to complete the following tasks in order to obtain an ABT as a SAS data set.

1. Define the ABT structure.

   In this step, you have to define ABT details. These details can include library name, subsetting capability (if any) that the ABT will require, and variables of the ABT. In addition, it can include information such as source tables or source columns and values of those columns that make up the ABT.

2. Build the ABT.

   Use the ABT definition to build the SAS data set with exactly the same structure that was given in the ABT definition.

In order to define the structures of ABTs for microsegmentation and offer ranking, you have to use template ABTs. The ABTs are automatically built when users run the microsegmentation and offer ranking workflow steps through the SAS Offer Optimization for Communications interface.

Prerequisites

Before you start building the template ABT as explained in the subsequent subsections, make sure that the following tasks are complete:

1. Populate the configuration area of the Analytics data mart that contains metadata of the base tables with appropriate data. For details, see "Populating the Configuration Area of Analytics Data Mart" on page 42.
2. Familiarize yourself with the contents of the ABT-specific workbooks. For details, see"About the ABT-Specific Workbook" on page 61.
3. Familiarize yourself with the recent behavioral, supplementary, and derived variables by reading through the Analytics data model. For details, see *SAS Offer Optimization for Communications: Analytics Data Mart*. For details about how to access this document, see **`http://support.sas.com/documentation/solutions/offeropt/index.html`**.
4. Refer to the code of the %cfdn_outer_macro.sas macro. This macro is located in the **`<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts`** folder. This macro contains the sample calls for all the macros that need to be run in order to create the ABTs. When you perform the instructions that are explained below, make sure that you have opened the %cfdn_outer_macro.sas macro. Also, the instructions specify the sequence in which these macros are to be run. Before you run the macros from this macro, make sure that you verify the parameter values that these macros require.

Acquire the Purpose-Specific ABT Configuration File

Make sure that you have downloaded the ABT-specific workbooks. For details, see "Download the Analytics Workbooks" on page 41. Use the correct workbook in order to define the template for the purpose-specific ABT.

Depending on the purpose for which you are defining the ABT template, you have to use the correct ABT configuration workbook.

Table 12.5 *Purpose-Specific ABT Configuration Workbooks*

Purpose	Purpose Short Code	Workbook Name
Microsegmentation postpaid	MSPOST	ABT_SPECIFIC_MS_postpaid.xls
Microsegmentation prepaid	MSPRE	ABT_SPECIFIC_MS_prepaid.xls
Offer ranking postpaid	ORPOST	ABT_SPECIFIC_OR_postpaid.xls
Offer ranking prepaid	ORPRE	ABT_SPECIFIC_OR_prepaid.xls

Note: This document refers to the ABT_SPECIFIC_MS_postpaid.xls workbook wherever necessary. However, you have to refer to the appropriate workbook according to the purpose for which you are defining the template ABT.

Create User in SAS Management Console

The user who executes the INSERT scripts that are mentioned in the instructions below needs access to the libraries that are predefined in the SAS environment. In order to enable the user to have this capability, make sure that a user is defined in SAS Management Console with the following role and group:

- Role: Metadata Server: Unrestricted
- Group: Communication Common Oracle User Group

Update the Name and Location of the Configuration File

To update the name and the location of the configuration file:

1. Go to the **<SAS configuration path>/Lev1/<SAS Application Server context name>** folder.
2. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
3. Open the %cfdn_outer_macro.sas file, which is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASOfferOptForCommServer5.2/SASCode/insertscripts** folder.
4. Add the following code:

   ```
   %let abtxls ="<path in which you have stored the purpose-sepcific
   workbook
   ```

   For example, you can add the following lines of code, if you are working with the ABT_SPECIFIC_MS_prepaid.xls workbook:

   ```
   %let abtxls ="C:/BPP/common/docs/docs_2/final/ABT_SPECIFIC_MS_postpa
   id.xls"
   ```

5. Click **Save**.
6. Run the line of code that you have added in step 4.
7. To ensure successful execution of the code, on the menu select **View** ⇨ **Log**.
8. Close Base SAS.

Set Up the Base SAS Environment

To set up the Base SAS environment for running the ABT-specific code:

1. Go to the **<SAS configuration path>/Lev1/<SAS Application Server context name>** folder.
2. Depending on whether the operating system is Windows or UNIX, run the sas.bat or the sas.sh file respectively. For example, on the Windows machine, run the **C:/SAS/Config/Lev1/SASOOC/sas.bat** file.
3. Make sure that you do not close Base SAS until you complete the configuration tasks. Otherwise, you have to perform steps 1 and 2 to set up the Base SAS environment.

Define the Population

When you create an analytical model, you have to define the population for which you are building the analytical model. The subset criteria feature of DABT enables you to define your population.

For a microsegmentation ABT, the list of customers that are retrieved after the target segment selection workflow step is run is the subset criteria.

Note: This task is applicable only for Microsegmentation workflow step.

To define the population:

1. Open the appropriate purpose-specific ABT workbook.
2. In the **subset_where** worksheet, search for the subset query that has the name "target segment" population. You have to use this query for microsegmentation.
3. Save and close the file.
4. In Base SAS, run the following macros from the %cfdn_outer_macro.sas macro:
   - %CFDN_SUBSETQUERY_IMPORT
   - %CFDN_CREATE_SUBSET_WRAPPER

   As a result, subset queries are entered into the SUBSET_QUERY_MASTER table of the Analytics data mart.

Create a DABT Project

You have to associate an ABT to a project. A project caters to only one ABT.

To create a DABT project:

1. Open the appropriate purpose-specific ABT workbook.
2. In the **Project** worksheet, specify the project details such as PROJECT_NM, LEVEL_NM, DESCRIPTION, LOCATION_PATH, QUERY_NM, and PURPOSE. For example, MSPOST_CUSTOMER is the default project name. You can change this information. However, do not change the entries.

   Note: Make sure that the QUERY_NM that you specify here is the same as the subset query that you retrieved in the **subset_where** worksheet when you define the population.
3. Save and close the file.
4. In Base SAS, run the following macros from the %cfdn_outer_macro.sas macro:
   - %CFDN_PROJECT_IMPORT
   - %CFDN_NEW_PROJECT_WRAPPER

   The PROJECT_MASTER table of the Analytics data mart is populated with the details of the new project.

Define ABT Details

This task considers the following assumptions:

- The type of ABT would always be template (indicated by value T of the ABT_TYPE_FLG) for any of the ABTs that are built for microsegmentation and offer ranking.
- The libref should be of the abtdata library to indicate that the ABT resides in the project-specific location that is pointed by the abtdata library. For example, if the call for building the ABT is made from a project called *project 1*, then the parameter abtdata would have the value **`<SAS configuration path>/Lev1/`**

Applications/SASFoundforComm5.2/Data/ooc_data/projectdata/prj1/abt . The ABT will be in this location after it is built.

To define the template ABT:

1. Open the appropriate purpose-specific ABT workbook.
2. In the **ABT** worksheet, provide details such as Table_Nm, Short_Nm, Description, Target_Time_Freq, Target_Time_Freq_Count, Library, LIBRARY_REFERENCE, LIBNAME_STATEMENT, Level, PROJECT_NM, FLG, Purpose, and ABT_TYPE. An ABT with the name **BPP_MSPOST_CUST** is provided in the **ABT** worksheet. Do not change any of the parameters except the ABT name.
3. Save and close the file.
4. In Base SAS, run the following macros from the .sas macro:
   - %CFDN_ABT_IMPORT
   - %CFDN_NEW_ABT_WRAPPER

   As a result, a new record of the modeling ABT that is to be created with respect to the DABT project is added in the ABT_MASTER table.

Define ABT Variables

To define variables for the modeling ABT:

1. Open the appropriate purpose-specific ABT workbook. For details, see "Acquire the Purpose-Specific ABT Configuration File" on page 181.
2. In the **beh** worksheet, specify appropriate values for the following columns:
   - Var_Group
   - ABT_Nm
   - Source_Table_Nm
   - Select_Column_Nm
   - Select_Column_Type_Cd
   - Select_Column_Value
3. Save and close the file.
4. In Base SAS, run the following macros from the %cfdn_outer_macro.sas. macro:
   a. Run the %CFDN_ABTVARS_IMPORT macro. This macro converts the data from the MS Excel file to a SAS data set.
   b. Run the %CFDN_ABT_DIMVAL_INSERT macro. This macro creates temporary SAS data sets for each type of variable that is mentioned in the macro.
   c. Run the %CFDN_CREATE_BEH_VAR_WRAPPER macro. This macro creates data sets for the behavioral variables in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/dabt_data/scratch/flex** folder. For example, if the abt_sk for which the ABT is being built is 15, then the beh_var_list_15.sas7bdat and beh_var_list_15_edit.sas7bdat data sets will be created in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Data/dabt_data/scratch/flex. beh_var_list_15_edit.sas7bdat** folder.

5. In the beh_var_list_15_edit.sas7bdat data set, verify the variable names. The values for the edit_cd and edit_option columns indicate whether the length of the variable names exceeds the maximum limit of 28 characters. Rename all the variables where edit_option is set to D (representing delete as the length is greater than 28 characters). Edit names of all such variables and change the edit_option for those to A to represent accept as the length is less than or equal to 28 characters.
6. In Base SAS, run the following macros from the %cfdn_outer_macro.sas. macro:
   a. Run the %CFDN_UPDATE_VAR_LIST macro to reflect the changes that are made in the non-editable data set. For example, it would be indicated by beh_var_list_15.sas7bdat.
   b. To save the variables, run the %CFDN_SAVE_BEH_VAR_WRAPPER macro. This step reads data from the temporary tables and enters data in the following tables:
      - VARIABLE_MASTER
      - BEHAVIORAL_VARIABLE
      - ABT_X_VARIABLE

Define Derived Variables

Before you create derived variables, make sure that you have complete information about them. For details, see "Derived Variables" on page 259.

To create derived variables:

1. Open the appropriate purpose-specific ABT workbook.
2. In the **derivedvardetails** and **derivedvarsource** worksheets, specify appropriate parameter values.
3. Save and close the file.
4. In Base SAS, Run the following macros from the %cfdn_outer_macro.sas macro:
   - %CFDN_DRVDVARS_IMPORT
   - %CFDN_SAVE_DRVD_VAR_WRAPPER

Register Variables in the Application Data Model

The ABTs for microsegmentation and offer ranking are built when users run these workflow steps. In order to integrate DABT and the SAS Offer Optimization for Communications interface, the template ABT variables that are defined by using the above steps have to be registered in the application data model. To do so, run the %cfdn_mod_abt_var_dts_wb macro. This macro populates the VARIABLE_ID, VARIABLE_NM, ANALYTICAL_VARIABLE_IND, VARIABLE_SUB_CTGRY_CD, VARIABLE_ID, CREATED_DTTM, VARIABLE_DISPLAY_IND, PREPAID_IND, POSTPAID_IND columns of the BPP_VRBL_MASTER table for all the variables of microsegmentation and offer ranking ABT.

In addition, perform the following tasks:

1. Specify the profiling code for the variables that will be used in the profiling step of microsegmentation as *NUM* in the VARIABLE_PRFLNG_TYPE_CD column of the BPP_VRBL_MASTER table manually.
2. (Optional) Define categories and subcategories for analytical variables.

   Note: By default, the subcategory for analytical variables is populated as **BEH** in the VARIABLE_SUB_CTGRY_CD column of the BPP_VRBL_MASTER table. However, if you want to further categorize these variables, then perform steps from 2a to 2e.

   a. Configure the categories for the analytical variables in the BPP_VARIABLE_CTGRY table.

   b. Specify the corresponding category values in the BPP_VARIABLE_CTGRY_NLS table.

   c. In the BPP_VARIABLE_SUB_CTGRY table, configure subcategories for the categories that you have defined in the BPP_VARIABLE_CTGRY table.

   d. Specify the corresponding subcategory values in the BPP_VARIABLE_SUB_CTGRY_NLS table.

   e. In the VARIABLE_SUB_CTGRY_CD column of the BPP_VRBL_MASTER table, specify the subcategories that you have defined in the BPP_VARIABLE_SUB_CTGRY table.
3. Configure the localized values in the BPP_VRBL_MASTER_NLS table. For details, see "Configure Variables" on page 166.

Configuring Parameters for Workflow Steps

The BPP_PARAMETERS_MASTER table contains parameters that you have to configure for SAS Offer Optimization for Communications. The values of these parameters are stored in the BPP_PARAMETERS_VAL_MASTER table. Each parameter has a default value. Default values are populated for all parameters. For some parameters, you must not change the default values. For these parameters, the **Is Editable** column in the table below is *No*. In addition, certain parameters can have multiple values. For these parameters, the **Multiple Values** column in the table below contains *Yes*.

Table 12.6 *Parameters for Workflow Steps*

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_FLAG_ON	indicates that the flag value for a certain variable is true.	No	No	Y	Y
BPP_FLAG_OFF	indicates that the flag value for a certain variable is false.	No	No	N	N

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_EXECUTION_MODE	indicates whether a project is in batch or design mode. This parameter is not used currently.	No	Yes	Y or N	N
BPP_SAMPLE_SIZE	indicates the sample size that has to be considered for building a clustering model in design mode. From the population that is filtered in the target segment, a sample of the size mentioned here is chosen. However, the size of the population that is filtered in the target segment can be less than the sample size that is configured here. In this case, the whole population will be considered for model building.	Yes	No	an integer value in between 40000 and 60000.	50000
BPP_VARCLUS_GROUP_NAME	used to filter data from the proc varclus output data set based on the `WHERE _type_ is to 'GROUP'` condition.	No	No	GROUP	GROUP
BPP_VARCLUS_RSQUARED_NAME	used to filter data from the proc varclus output data set based on the `WHERE _type_ is equal to 'RSQUAR` condition.	No	No	RSQUARED	RSQUARED

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_NUM_RECOMMENDED	determines the default number of analytically recommended variables that are selected from each variable cluster group.	Yes	No	an integer that is greater than or equal to 1. However, it is recommended that you set a value that is less than 4.	2
BPP_NUM_PROFILE	determines the default number of profiling variables that are selected from each variable cluster group.	Yes	No	an integer that is greater than or equal to the value of the BPP_NUM_RECOMMENDED parameter. However, it is recommended that you set a value that is less than 5.	2
BPP_ELBOW_CRIT	determines the default value for the elbow criterion in the clustering procedure.	Yes	No	any numeric value between 0 and 1.	0.01
BPP_CONVERGENCE_CRIT	determines the default value for the convergence criterion in the clustering procedure.	Yes	No	any nonnegative value.	0.01
BPP_MAXCLUSTER	determines the default value for the maximum number of clusters.	Yes	No	any numeric value between 2 and 999.	5
BPP_MS_RSQUARE	used to filter data from the proc fastclus output data set based on the `WHERE _type_ is to 'RSQ'` condition.	No	No	RSQ	RSQ

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_MS_MAXITER	determines the default value for maximum number of iterations in the clustering procedure.	Yes	No	an integer that is greater than 1.	500
BPP_OUTLIER_FLG_NAME	used to compare the outlier flag in the BPP_STATISTICS table. This parameter is not used currently.	No	No	OUTLIER_FLG	OUTLIER_FLG
BPP_FREQUENCY_NAME	compares frequency in the BPP_STATISTICS table. This parameter is not used currently.	No	No	FREQ	FREQ
BPP_DISPERSION_NAME	compares dispersion in the BPP_statistics table. This parameter is not used currently.	No	No	DISPERSION	DISPERSION
BPP_P_VAL_CUT_OFF	compares calculated confidence value with theoretical confidence value in all analytical processes. Confidence value = 1 - P value.	Yes	No	0.95	0.95
BPP_CATEGORICAL_VARIABLE_TYPE	identifies whether a variable is of the categorical type.	No	No	CAT	CAT
BPP_NUMERICAL_VARIABLE_TYPE	identifies whether a variable is of the numerical type.	No	No	NUM	NUM

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_PROFILE_ASSIST_ANOVA_NAME	used to filter data from the proc anova output data set based on the `WHERE _type_ is to 'ANOVA'` condition.	No	No	ANOVA	ANOVA
BPP_SEGMENT_SCORE_CENTER_NAME	used to filter data in the segmentation scoring method.	No	No	CENTER	CENTER
BPP_PERCENT_OUTLIER	used to select the outermost population from centers in outlier handling process for calculation.	Yes	No	any value between 0 and 1.	0.01
BPP_PCT_PERMTD_CENTROID_SHIFT	used to check the deviation in shifted centroid and original centroid in outlier handling process.	Yes	No	any value greater than 0.	0.001
BPP_OUTLIER_HANDLE_CENTER_NAME	used to filter data in outlier handling process.	No	No	CENTER	CENTER
BPP_OUTLIER_HANDLE_MEAN_NAME	used to filter data from the proc means output data set based on the `WHERE _type_ is to 'MEAN'` condition.	No	No	MEAN	MEAN
BPP_SHIFTED_CENTER_NAME	used to filter data in outlier handling process.	No	No	SHIFTED_CENTER	SHIFTED_CENTER
BPP_SAMPLING_METHOD	defines list of values for the sampling method.	No	Yes	CENTROID and SPREADBASED	CENTROID

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_OFFERS_PER_MICROSEGMENT	contains the default value for top number of offers for the offer ranking at microsegment level workflow step.	Yes	No	an integer that is greater than or equal to 1.	10
BPP_OFFERS_PER_CUSTOMER	contains the default value for top number of offers for the offer ranking at customer level workflow step.	Yes	No	an integer that is greater than or equal to 1.	5
BPP_PERMITTED_VARIATION_PER_DIST	determines the deviation in distribution between microsegments for model monitoring process.	Yes	No	any value that is greater than 0.	5
BPP_PERMITTED_VARIATION_STD_DEV	determines the deviation in dispersion between microsegments for model monitoring process.	Yes	No	any value that is greater than 0.	0.5
BPP_RECALIBRATION_DIST_REASON	indicates that a model is recalibrated for the reason of distribution of microsegments.	No	No	D	D
BPP_RECALIBRATION_STD_REASON	indicates that a model is to be recalibrated for the reason of dispersion in microsegment.	No	No	S	S

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_RECALIBRATION_BOTH_REASON	indicates that a model is to be recalibrated for the reason of dispersion and distribution in microsegment.	No	No	B	B
BPP_RECALIBRATION_NO_REASON	used if no reason is available for recalibration.	No	No	N	N
BPP_ELIGIBILITY_SUPPORT_FLG	used to handle eligibility criteria.	No	Yes	Y or N.	N
BPP_ERR_THRESHOLD	the threshold for critical error.	No	No	4	4
BPP_OPT_CRIT	used to determine the list of values for optimization criteria in offer ranking.	No	Yes	MIN and MAX	MIN
BPP_OPT_WT	contains the default value for optimization weight in offer ranking.	Yes	No	any integer that is greater than or equal to 1	10
BPP_OPT_DEVIATION	contains the default value for optimization deviation in offer ranking.	Yes	No	an integer that is greater than or equal to 1.	3
BPP_OPT_MSNGVAL	is used to define list of values for the missing value replacement methods in offer ranking.	No	Yes	MIN, MAX, and AVERAGE	MIN
BPP_LANGUAGE_CD	the language code that is used in the Foundation data mart.	Yes	No	Standard codes for various languages. For example, en_US for US English.	en_US

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_PREPAID_OFFR_PYMNT_MODE_CD	the offer payment code for prepaid as defined in the Foundation data mart.	Yes	No	should be same as the offer payment code for prepaid that is defined in the Foundation data mart.	PREPAID
BPP_POSTPAID_OFFR_PYMNT_MODE_CD	the offer payment code for postpaid as defined in the Foundation data mart.	Yes	No	should be same as the offer payment code for postpaid that is defined in the Foundation data mart.	POSTPAID
BPP_MSPRE	used in ABT creation step of microsegment and offer payment code of prepaid.	No	No	MSPRE	MSPRE
BPP_MSPOST	used in ABT creation step of microsegment and offer payment code of postpaid.	No	No	MSPOST	MSPOST
BPP_INC_NEW_OFRBNDL_IND	contains the default value for the indicator of new offer bundles that will be included in offer assembly.	Yes	No	Y or N	Y
BPP_FLT_ASSIGNED_PROD_IND	indicates whether the offer evaluation should check for bundles that include only the existing representative customer-assigned product or also bundles that contains additional assigned products	Yes	No	Y or N	Y

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_FLT_ACC_TO_USG_IND	used to set default value for the indicator that determines whether bundles that include rating schema for the existing representative customer usage should be considered.	Yes	No	Y or N	Y
BPP_NUM_OF_CYCLES	contains the default value for number of cycles that are to be considered in invoice recalculation.	Yes	No	an integer that is greater than or equal to 1	3
BPP_SESSION_ID	used to distinguish between different interface sessions.	No	No	0	0
BPP_SLEEP_DURATION	used to halt the processes for a given time while polling for response in interface sessions. It is in seconds.	Yes	No	an integer greater than or equal to 1	30
BPP_DB_NM	used to check the current database name.	No	No	ORACLE	ORACLE
BPP_TIMEOUT	used to terminate the processes after the given time while polling for response in interface sessions. It is in seconds.	Yes	No	any integer greater than or equal to 1	180

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_CUST_BILL_AMT_VAR	indicates that the net bill amount variable is defined in the variable master table.	No	No	SMS	SMS
BPP_SMS	filters data based on short message service.	Yes	No	should be equal to the service name associated with the short message service.	1_SERVICE_NM
BPP_MMS	filters data based on multimedia message service.	Yes	No	should be equal to the service name associated with the multimedia message service.	21_SERVICE_NM
BPP_PRJ_SCHLD_FRQNCY	used to show possible schedule frequencies for batch execution.	Yes	Yes	MONTHLY, WEEKLY and DAILY	MONTHLY
BPP_ORPRE	is used in ABT creation step of offer ranking and offer payment code of postpaid.	No	No	ORPRE	ORPRE
BPP_ORPOST	used in ABT creation step of offer ranking and offer payment code of postpaid.	No	No	ORPOST	ORPOST
BPP_PAYMENT_MODE	used to handle payment-mode-specific execution.	No	Yes	PREPAID and POSTPAID	POSTPAID
BPP_PREPAID_MODE_CD	determines whether the payment mode is prepaid.	No	No	PREPAID	PREPAID
BPP_POSTPAID_MODE_CD	determines whether the payment mode is postpaid.	No	No	POSTPAID	POSTPAID
BPP_LOYALTY_LEVEL_CD	defines the loyalty level code.	No	No	CUST	CUST

Parameter Name	Description	Is Editable	Multiple Values	Possible Values	Default Value
BPP_ABTSCORE_FLG	used in ABT creation to compare the scoring flag.	No	No	S	S
BPP_AGRMNT_LVL_NUM	contains the agreement level number.	No	No	1	1
BPP_CENTROID	determines whether the sampling method is centroid.	No	No	CENTROID	CENTROID
BPP_SPREADBASED	determines whether the sampling method is spread-based.	No	No	SPREADBASED	SPREADBASED
BPP_NORM_MEAN	contains the default mean value for normalization procedure.	Yes	No	any value	0
BPP_NORM_STD	contains the default value for standard deviation for normalization procedure.	Yes	No	any value greater than 0	1
BPP_OPEN_CYCLE_IND	is used to calculate invoice recalculation with open cycle end.	No	No	Y	Y
BPP_DATEFUNC	is used to calculate the datepart from datetime format.	No	No	datepart	datepart
BPP_SEED_DELETE_VAL	is used in clustering procedure to restrict the minimum cluster size.	Yes	No	an integer that is greater than 1	4
BPP_IMPUTE_METHOD	is used for replacing missing values of an ABT	Yes	Yes	MIN, MAX, MEAN, and NONE	MIN

Prerequisite Tasks for Creating Projects

Overview

After you configure variables and values that are required for various workflow steps and define business groups, you have to populate the BPP_VRBL_X_WKFLWSTP and WKFLWSTP_AVBL_VRBL_VAL_MASTER tables. You must populate these tables in order to ensure smooth functioning of the workflow steps.

Note: You have to populate both the tables before you define projects and configure the workflow. Otherwise, the workflow steps will not execute successfully.

Populate the BPP_VRBL_X_WKFLWSTP Table

The BPP_VRBL_X_WKFLWSTP table contains information about variables that are required for processing workflow steps such as target segment selection, microsegmentation, and offer ranking. For each combination of a business group and a workflow step, you can define a set of variables. Therefore, all projects that are associated with a particular business group will have the same set of variables.

Note: The analytical variables that are part of microsegmentation and offer ranking ABT must also be configured when you perform this step. For details, see "Register Variables in the Application Data Model" on page 185.

To populate the BPP_VRBL_X_WKFLWSTP table, specify the following information:

Table 12.7 *Columns of BPP_VRBL_X_WKFLWSTP Table*

Column	Description
VARIABLE_ID	the sequence number of the variable as specified in the BPP_VRBL_MASTER table.
WORKFLOW_STEP_ID	the sequence number of the workflow step as specified in the BPP_WKFLWSTP_MASTER table.
BSNSGRP_ID	the unique identifier of the business group for whose projects you want to configure the variables. This business group ID is as generated in the BSNSGRP_SUMMARY table.

Populate the WKFLWSTP_AVBL_VRBL_VAL_MASTER Table

You have to populate the WKFLWSTP_AVBL_VRBL_VAL_MASTER table to identify the list of values that you want to provide to users. Users will use these values to define filter criteria for the target segment selection workflow step. For each combination of a business group and a workflow step, you can define the values that you want to consider for defining target segments that are associated with a particular

business group. To populate the WKFLWSTP_AVBL_VRBL_VAL_MASTER table, specify the following information.

Table 12.8 *Columns of WKFLWSTP_AVBL_VRBL_VAL_MASTER Table*

Column	Description
BPP_REF_VARIABLE_VALUE_ID	the sequence number of the reference value that is specified in the BPP_REF_VRBL_VAL_MASTER table.
WORKFLOW_STEP_ID	the sequence number of the workflow step as specified in the BPP_WKFLWSTP_MASTER table.
BSNSGRP_ID	the unique identifier of the business group for whose projects you want to configure the variables. This business group ID is as generated in the BSNSGRP_SUMMARY table.

External Interfaces

Overview

SAS Offer Optimization for Communications interfaces with external source systems in order to exchange data for Offer Assembly and Invoice Recalculation workflow steps.

In the Microsegment Representation workflow step, representative customers are drawn. In the Offer Assembly workflow step, the information about representative customers is sent to the external source system. As a result, SAS Offer Optimization for Communications receives information about eligible offers for these customers from the external source system .

In the Invoice Recalculation step, information about each combination of the representative customer and the eligible offer is sent to the external source system. As a result, SAS Offer Optimization for Communications receives billing information for each combination from the external source system.

For details about the workflow steps, see *SAS Offer Optimization for Communications: User's Guide*.

Configuring the Setup for External Interfaces

Populating Interface Tables

Before users run Offer Assembly and Invoice Recalculation workflow steps, you have to configure certain interface tables. For details about the interface tables, see *SAS Offer Optimization for Communications: Data Dictionary*.

For the following interface tables, data is populated when you perform the post-installation tasks:

- BPP_INF_PRM
- BPP_INF_STATUS
- BPP_INF_STATUS_NLS

Therefore, make sure that you do not change data in these tables.

Configure Error Codes

In order to track the errors that occur during data processing and data exchange between SAS Offer Optimization for Communications and the external source system, you have to define certain error codes.

To support error tracking:

1. Define error codes in the BPP_INF_ERROR table.
2. Define localized display names and descriptions along with locale and error codes in the BPP_INF_ERROR_NLS table.
3. Define error codes in the bpp_smd_error.smd file.

Configure Parameters

When SAS Offer Optimization for Communications interfaces with an external source system, some data is populated into the interface tables. Data is populated into the interface tables based on which workflow step (Offer Assembly or Invoice Recalculation) is being processed by the SAS code. After data is populated into the interface tables, the back-end processes go into a sleep mode until they receive a response from the external source system.

To ensure successful execution of Offer Assembly and Invoice Recalculation workflow steps, you have to set correct values for the following parameters.

BPP_TIMEOUT
: the total duration for which back-end processes wait to receive response from the external interface. If there is no response in this time span, the back-end processes stop their execution. For example, you set the BPP_TIMEOUT duration as 1 hour. If the back-end processes do not receive any response from the external interface within an hour after data is populated into the interface tables, the processes will stop their execution.

BPP_SLEEP_DURATION
: the duration, which is used to set the polling frequency. After data is loaded into the interface tables, back-end processes start polling to check the response from the external interface at regular time intervals. For example, if you set the BPP_SLEEP_DURATION as two minutes, then the back-end processes check the response after every two minutes in the BPP_TIMEOUT period.

Data Processing Steps

Overview

The following steps explain how data is exchanged between SAS Offer Optimization for Communications and the external interface when users run the Offer Assembly and Invoice Recalculation workflow steps. At various steps, the audit table BPP_INF_AUDIT is updated. For details, see “Audit Table” on page 201.

Offer Assembly Workflow Step

When users run the Offer Assembly workflow step, the following back-end processes are run.

1. Creation of unique session ID for the current session.
2. The back-end process inserts details about parameters that are related to the Offer Assembly workflow step into the BPP_INF_PRM_DTL interface table with the current session ID.
3. The back-end process inserts details about representative customers that are drawn in the Microsegment Representation workflow step into the BPP_INF_OFFER_ASSMBLY_REP_CUST interface table with the current session ID.
4. The back-end process makes an entry in the BPP_INF_AUDIT interface table with the current session ID and starts polling for the response from the external interface.
5. The data from the external source system is populated in the BPP_INF_AUDIT and BPP_INF_OFFER_ASSMBLY_DTL interface tables.
6. The details about eligible offers for all representative customers from the external source system are populated in the BPP_INF_OFFER_ASSMBLY_DTL interface table with the current session ID.
7. If the external source system sends a Successful status in the BPP_INF_AUDIT interface table, then the back-end process copies data from the BPP_INF_OFFER_ASSMBLY_DTL interface table into the OFFER_ASSMBLY_DTL application data mart table based on the current session ID.
8. If the external source system sends a Failed status in the BPP_INF_AUDIT interface table, then the back-end process does not copy any data from interface table to application data mart table and marks the status of Offer Assembly workflow step as Failed.
9. If no response is received from the external interface within the configured time, then the back-end process stops processing and marks the status of the Offer Assembly workflow step as Failed.

Invoice Recalculation Workflow Step

When users run the Invoice Recalculation workflow step, the following back-end processes are run.

1. Creation of unique session ID for the current session.
2. The back-end process inserts details about parameters that are related to the Invoice Recalculation workflow step into the BPP_INF_PRM_DTL interface table with the current session ID.
3. The back-end process inserts details about eligible offers for all representative customers into the BPP_INF_BILLRECALC_INPUT interface table with the current session ID.
4. The back-end process makes an entry into the BPP_INF_AUDIT interface table with the current session ID and starts polling for the response from the external source system.
5. The output data is populated in the BPP_INF_AUDIT, BPP_INF_BILLRECALC_DTL, BPP_INF_BILLRECALC_USG_DTL, and BPP_INF_BILLRECALC_NONUSG_DTL interface tables.
6. The following details are populated in the interface tables:

BPP_INF_BILLRECALC_USG_DTL
: billing details about usage at product-event type grain on each eligible offer for all representative customers are populated with current session ID. The time grain of the table is same as the billing cycle.

BPP_INF_BILLRECALC_NONUSG_DTL
: billing details of non-usage at product-charge type grain on each eligible offer for all representative customers are populated with current session ID. The time grain of the table is same as the billing cycle.

BPP_INF_BILLRECALC_DTL
: the summarized bill details about each eligible offer for all representative customers are populated with the current session ID. The time grain of the table is same as the billing cycle.

7. If the external source system sends a Successful status in the BPP_INF_AUDIT interface table, then the back-end process copies data from the BPP_INF_BILLRECALC_USG_DTL, BPP_INF_BILLRECALC_NONUSG_DTL, and BPP_INF_BILLRECALC_DTL interface tables to the BILLRECALC_USG_DTL, BILLRECALC_NONUSG_DTL, and BILLRECALC_DTL application data mart tables, respectively, based on the current session ID.
8. If the external source system sends a Failed status into the BPP_INF_AUDIT interface table, then the back-end process does not copy any data from interface tables to application data mart tables and marks the status of the Invoice Recalculation workflow step as Failed.
9. If no response is received from the external source system within the configured time, then the back-end process stops processing and marks the status of the Invoice Recalculation step as Failed.

Audit Table

All activities in Offer Assembly and Invoice Recalculation are tracked in the Audit table. The status of each activity is tracked through unique codes.

Table 12.9 *Status Codes for Offer Assembly*

Sequence	Interaction Code	External Source System Code	Status Code	Populated by	Action
1	RQ	BI	ENBOF	SAS Offer Optimization for Communications	When SAS Offer Optimization for Communications populates data for offer assembly, it inserts data for interaction code RQ.

Sequence	Interaction Code	External Source System Code	Status Code	Populated by	Action
2	RQ	OF	OFPRS	External source system	When the external source system starts processing for offer assembly, it updates the source system code and status code for the interaction code RQ.
3	RQ	OF	OFEXS or OFEWE	External source system	When the external source system interface completes processing successfully or with errors for offer assembly, it updates the source system code and status code for the interaction code RQ.
4	RS	OF	ENBBIOF	External source system	When the external source system completes processing successfully or with errors for offer assembly, it will insert data for the interaction code RS.

Table 12.10 *Status Codes for Invoice Recalculation*

Sequence	Interaction Code	External Source System Code	Status Code	Populated by	Action
1	RQ	BI	ENBBC	SAS Offer Optimization for Communications	When SAS Offer Optimization for Communications populates data for Invoice Recalculation process, it inserts data for the Interaction code RQ.

2	RQ	BC	BCPRS	External source system	When external interface starts processing for Invoice Recalculation, it will update the Source System code and the Status code for the Interaction code RQ.
3	RQ	BC	BCSU1/BCEWE	External source system	When external source system completes processing successfully or ended with error for Invoice Recalculation, it updates the Source System code and the status code for the interaction code RQ.
4	RS	BC	ENBBIBC	External source system	When the external source system completes processing successfully or with errors for Invoice Recalculation, it inserts data for the interaction code RS.

Configuring Error Tables for Workflow Steps

Overview

Errors can occur when you run workflow steps from the SAS Offer Optimization for Communications interface. In order to enable users to resolve the errors, an appropriate error message needs to be displayed. In addition, these error messages should be included in the SAS log. To complete these tasks, you have to configure the error tables.

Create an SMD Data Set

To create an SMD data set from an smd file:

1. Update the textual content for all error codes in an smd file.

2. Rename the smd file to its original name with a suffix according to the language code.
3. Run the bpp_create_smd_to_ds.sas macro. You can run this macro from SAS Enterprise Guide. This macro requires the following inputs:

   Dir
   the name of the directory in which the smd file exists.

   Smdfilename
   the original smd filename.

   Localelist
   the locale for which you want to add error codes.

   Outputlib
   the directory in which you want to create the smd data set.

   TIP Repeat these steps for all the smd files.

Configure Error Codes

In order to display appropriate messages in the SAS Offer Optimization for Communication interface, add localized descriptions for all error codes in the BPP_ERR_NLS table.

Table 12.11 *BPP_ERR_NLS Table*

Column	Description
ERR_CD	the error code that is used in the SAS code to identify the reason of failure.
LOCALE	a specific language code that is associated with a geographical region.
ERR_DSPLY_DESC	the localized display description that is associated with the error code.

Creating Projects

After you complete the configuration setup for the projects, workflow steps, and external interfaces, users can define projects and configure workflow steps using the SAS Offer Optimization for Communications interface. For details, see *SAS Offer Optimization for Communications: User's Guide*.

Logs for Workflow Steps

When you run or reset a workflow step through the SAS Offer Optimization for Communications interface, a log file is created in the **<SAS configuration path>/**

Lev1/Applications/SASOfferOptForCommServer5.2/Logs/ooc_logs/oocprojectlogs folder. This log file contains information about all the tasks and the errors that might occur when you perform these tasks. For example, if you run the Target Segment Selection workflow step for a project with ID 3, then the bpp_spexecutetargetsegment_3.log file is generated. Also, when you reset the Target Segment Selection workflow step for a project with ID 3, then the bpp_spresettargetsegment_3.log file is generated. Similarly, log files are generated when you run and reset other workflow steps through the SAS Offer Optimization for Communications interface.

Chapter 13

Configuration for Workflow Reports

Configuring Setup for Workflow Reports

Overview

SAS Offer Optimization for Communications enables users to define dynamic reports at workflow step level. Based on these reports, users can evaluate the results that are derived at various workflow steps and can decide whether they need to configure a workflow step again. These reports are called workflow reports.

Here is the list of the major features of workflow reports:

- You can define dimensions, measures, and derived measures dynamically.
- You can use various types of aggregations such as SUM, COUNT, MIN, and MAX to create measures.
- You can create workflow reports in graphical formats such as pie charts and bar charts. You can also generate these reports as data tables.

In order to enable users to define workflow reports, you have to complete the following configuration tasks:

1. Configure report categories.
    a. Associate a report category with one or more source report tables.
    b. Associate a report category with one or more workflow steps.
2. Configure reporting variables.
    a. Categorize reporting variables as dimensions, measures, and derived measures.

b. Configure a simple measure.
c. Associate measures with reporting variables.
d. Configure a derived measure.
e. Associate derived variables with simple measures.

3. Associate reporting categories with reporting variables.

Configuring Report Categories

Define a Report Category

A report category enables users to logically group reporting variables based on which they can further define reports. Therefore, a report category helps you ensure that a correct set of variables is grouped together. You have to provide a set of predefined report categories to users. To do so, you have to populate appropriate data in the BPP_RPT_CTGRY table.

Table 13.1 *BPP_RPT_CTGRY Table*

Column	Description
RPT_CTGRY_ID	the unique sequential number that is assigned to a report category.
RPT_CTGRY_NM	the name of the report category.
RPT_CTGRY_LEVEL_CD	the code for the level that is associated with the grain of report category. For example, the grain of report category can be at offer or product level.
RPT_CTGRY_DISPLAY_IND	indicates whether the report category is to be displayed in the SAS Offer Optimization for Communications interface. This column can have a value Y or N.
RPT_CTGRY_VALID_IND	indicates whether the report category is valid. This column can have a value Y or N.
AVLB_DYNMC_RPT_IND	indicates whether the report category is available for dynamic reporting. This column can have a value Y or N.

Example: Configure BPP_RPT_CTGRY Table

In order to define a report category that consists of usage revenue contribution across microsegments, insert the following values into BPP_RPT_CTGRY table.

Table 13.2 *Sample Data in the BPP_RPT_CTGRY Table*

Column	Value
RPT_CTGRY_ID	2

Column	Value
RPT_CTGRY_NM	USG_RVN_CNTRBTN_ACRSS_SGMNT_OFFR_LEVEL
RPT_CTGRY_LEVEL_CD	Y
RPT_CTGRY_DISPLAY_IND	Y
RPT_CTGRY_VALID_IND	OFFER
AVLB_DYNMC_RPT_IND	Y

After you configure categories in the BPP_RPT_CTGRY table, make sure that you enter correct values for these categories in the BPP_RPT_CTGRY_NLS table. To do so, you have to populate appropriate data in the BPP_RPT_CTGRY_NLS table.

Table 13.3 *BPP_RPT_CTGRY_NLS Table*

Column	Description
RPT_CTGRY_ID	the unique sequential number that is assigned to a report category.
LOCALE	identifies a specific language code that is associated with a geographical region.
RPT_CTGRY_DSPLY_NM	the localized display name that is associated with the category.
RPT_CTGRY_DSPLY_DESC	the localized display description that is associated with the category.

Example: Configure BPP_RPT_CTGRY_NLS Table

After you configure values for the USG_RVN_CNTRBTN_ACRSS_SGMNT_OFFR_LEVEL category into the BPP_RPT_CTGRY table, insert the following data into the BPP_RPT_CTGRY_NLS table to complete its localized configuration.

Table 13.4 *Sample Data in BPP_RPT_CTGRY_NLS Table*

Column	Description
RPT_CTGRY_ID	2
LOCALE	en_US
RPT_CTGRY_DSPLY_NM	USAGE REVENUE CONTRIBUTION ACROSS SEGMENT OFFER LEVEL
RPT_CTGRY_DSPLY_DESC	USAGE REVENUE CONTRIBUTION ACROSS SEGMENT OFFER LEVEL

Associate a Report Category with a Report Source Table

The values for reporting variables are retrieved from the report source table. Data is populated in these source tables through predefined ETL processes. The following report source tables are available:

- CUST_MONTHLY_SUMMARY_RF
- CUST_SERVICE_MONTHLY_RF
- CUST_SERVICE_WEEKLY_RF
- CUST_WEEKLY_SUMMARY_RF

A report category can use different report source tables in order to retrieve values for the reporting variables. However, for a combination of report category and payment mode, there can be only a single source table. Therefore, values for reporting variables are retrieved from the corresponding table based on the payment mode of the project and the report category that the user is defining in the workflow report. In order to associate a report category with a report source table, you have to populate appropriate data in the BPP_RPT_CTGRY_SRCTBL table.

***Table 13.5** BPP_RPT_CTGRY_SRCTBL Table*

Column	Description
RPT_CTGRY_ID	the unique identifier for a report category that is defined in the BPP_RPT_CTGRY table.
RPT_SRC_TABLE_NM	the name of the table that sources the population of reporting variables. For example, for postpaid the CUST_MONTHLY_SUMMARY_RF source table populates the reporting variables. However, for prepaid the CUST_WEEKLY_SUMMARY_RF source table populates the reporting variables.
BPP_PRJ_PYMNT_MODE_CD	the code for the payment mode that is associated with the target population for a project. The value that you configure for this column must be the same as the value that you have configured for the BPP_PAYMENT_MODE parameter in the BPP_PARAMETERS_VALUE_MASTER table. For example, the values that can be configured can be prepaid and postpaid.

Example: Configure BPP_RPT_CTGRY_SRCTBL Table

In order to map source tables with the USG_RVN_CNTRBTN_ACRSS_SGMNT_OFFR_LEVEL category, insert the following data into the BPP_RPT_CTGRY_SRCTBL table.

Table 13.6 *Sample Data in BPP_RPT_CTGRY_SRCTBL Table*

Column	RPT_CTGRY_ID	BPP_PRJ_PYMNT_MODE_CD	RPT_SRC_TABLE_NM
Record 1	2	POSTPAID	CUST_MONTHLY_SUMMARY_RF
Record 2	2	PREPAID	CUST_WEEKLY_SUMMARY_RF

Associate a Report Category with a Workflow Step

You have to configure the BPP_RPT_CTGRY_X_WKFLWSTP table in order to define valid combinations of report categories and workflow steps. Based on the combinations that you define, users can define workflow reports for various workflow steps.

Table 13.7 *BPP_RPT_CTGRY_X_WKFLWSTP Table*

Column	Description
RPT_CTGRY_ID	the unique identifier for a report category.
WORKFLOW_STEP_ID	the unique identifier that is assigned to a workflow step.

Example: Configure BPP_RPT_CTGRY_X_WKFLWSTP Table

To define the scope for the USG_RVN_CNTRBTN_ACRSS_SGMNT_OFFR_LEVEL category within workflow steps, insert the following data into the BPP_RPT_CTGRY_X_WKFLWSTP table.

Table 13.8 *Sample Data in BPP_RPT_CTGRY_X_WKFLWSTP Table*

Column	Value
RPT_CTGRY_ID	2
WORKFLOW_STEP_ID	3

Configure Reporting Variables

Categorize a Reporting Variable

In order to enable users to generate workflow reports based on a set of predefined reporting variables, you have to configure the BPP_RPT_VRBL_MASTER table. Users can define workflow reports based on the reporting variables that you configure in this table. A variable can be identified based on the following categories:

Dimensions
: A categorical variable can be categorized as a dimension. A churn band variable can be an example of a dimension.

Measures
A numerical variable that is based on simple aggregation.

Derived Measure
A numerical variable that is based on the calculation of two or more measures.

Table 13.9 *BPP_RPT_VRBL_MASTER Table*

Column	Description
VARIABLE_ID	the unique identifier for the reporting variable.
VARIABLE_NM	the name of the variable.
VARIABLE_TYPE_CD	the code that is assigned for the category of the variable. For example, the supplied values can be DIM, MSR, or DRVDMSR.
VARIABLE_DATA_TYPE	the data type of the variable. Based on the value that you specify for this column, the variable is displayed in appropriate format in the workflow reports.
VARIABLE_DISPLAY_FORMAT	the format in which a variable can be displayed in the SAS Offer Optimization for Communications interface. The supplied values can be currency, amount, data, and so on.
VARIABLE_DISPLAY_IND	indicates whether the variable will be displayed in the SAS Offer Optimization for Communications interface.
RPT_SRC_COLUMN_NM	the name of the column that sources the reporting variable.
RPT_SRC_LBRY_NM	the name of the library that sources the population of reporting variables
RPT_SRC_TABLE_NM	the name of the table that sources the population of reporting variables.
UPDATED_DTTM	the date and time that a record was updated.
CREATED_DTTM	the date and time that a record was created.

Example: Configure Derived Measure

You have configured dimensions, measures, and derived measures in the BPP_RPT_VRBL_MASTER table. To configure AVG_MOU_PER_CALL as a derived measure, define TOT_CALLS and TOT_MOU as measures in the BPP_RPT_VRBL_MASTER table. Also, define NUM_OF_CALLS and DURATION_OF_CALLS as sources of measures in the BPP_RPT_VRBL_MASTER table. To configure all these variables, insert the following data into the BPP_RPT_VRBL_MASTER table.

Table 13.10 *Sample Data in BPP_RPT_VRBL_MASTER Table*

Column	VARIABLE_ID	VARIABLE_NM	VARIABLE_DISPLAY_IND	RPT_SRC_TABLE_NM	VARIABLE_DISPLAY_FORMAT	VARIABLE_DATA_TYPE	VARIABLE_TYPE_CD	RPT_SRC_COLUMN_NM	RPT_SRC_LBRY_NM
Record 1	22	NUM_OF_CALLS	N	Null	NUMBER	NUMERIC(10,0)	MSRSRC	NUM_OF_CALLS	BPPAPCFG
Record 2	23	DURATION_OF_CALLS	N	Null	NUMBER	NUMERIC(10,0)	MSRSRC	DURATION_OF_CALLS	BPPAPCFG
Record 3	44	TOT_CALLS	Y	Null	NUMBER	NUMERIC(10,0)	MSR	Null	BPPAPCFG
Record 4	45	TOT_MOU	Y	Null	NUMBER	NUMERIC(10,0)	MSR	Null	BPPAPCFG
Record 5	64	AVG_MOU_PER_CALL	Y	Null	NUMBER	NUMERIC(15,3)	DRVDMSR	Null	BPPAPCFG

The CREATED_DTTM and UPDATED_DTTM columns hold values of the current timestamp.

To ensure that the correct display name of dimensions, measures and derived measures in the SAS Offer Optimization for Communication interface, configure the BPP_RPT_VRBL_MASTER_NLS table.

Table 13.11 *BPP_RPT_VRBL_MASTER_NLS Table*

Column	Description
VARIABLE_ID	the unique identifier for the reporting variable.
LOCALE	identifies a specific language code that is associated with a geographical region.
VARIABLE_DSPLY_NM	the localized display name that is associated with the variable.
VARIABLE_DSPLY_DESC	the localized display description that is associated with the variable.

Example: Configure BPP_RPT_VRBL_MASTER_NLS Table

To complete the configuration of variables, add the following data into the BPP_RPT_VRBL_MASTER_NLS table.

Table 13.12 Sample Data in BPP_RPT_VRBL_MASTER_NLS Table

Column	VARIABLE_ID	LOCALE	VARIABLE_DSPLY_NM	VARIABLE_DSPLY_DESC
Record 1	23	en_US	DURATION OF CALLS	DURATION OF CALLS
Record 2	24	en_US	NUMBER OF MESSAGES	NUMBER OF MESSAGE
Record 3	44	en_US	TOTAL CALLS	TOTAL CALLS
Record 4	45	en_US	TOTAL MOU	TOTAL MOU
Record 5	64	en_US	AVERAGE MOU PER CALL	AVERAGE MOU PER CALL

Configure a Simple Measure

You have to configure the measures that you want to include in the workflow reports in the BPP_RPT_MEASURE_MASTER table. These measures can be further used to define derived measures. In order to define a simple measure, specify appropriate values for the following columns of the BPP_RPT_MEASURE_MASTER table.

Table 13.13 BPP_RPT_MEASURE_MASTER Table

Column	Description
MEASURE_ID	the unique identifier for a variable that is defined as a measure for workflow reports. The value for this column must be numeric.
MEASURE_NM	the name of the variable that is defined as a measure in workflow reports.
PRMRY_AGGR_TYPE_ID	the unique identifier for primary type of aggregation that is performed on a measure. The BPP_RPT_AGGR_MASTER table contains the list of aggregations.
SCNDRY_AGGR_TYPE_ID	the unique identifier for any secondary type of aggregation that is performed on a measure. The BPP_RPT_AGGR_MASTER table contains the list of aggregations.
MEASURE_CALC_CNDTN_EXPR	This column is not used currently.

Example: Configure BPP_RPT_MEASURE_MASTER Table

You have defined measures in the BPP_RPT_VRBL_MASTER table in the previous step. Now, you have to define their properties into the BPP_RPT_MEASURE_MASTER table. Add the following data into BPP_RPT_MEASURE_MASTER table.

Table 13.14 *Sample Data in BPP_RPT_MEASURE_MASTER Table*

Column	MEASURE_ID	MEASURE_NM	PRMRY_AGGR_TYPE_ID	SCNDRY_AGGR_TYPE_ID	MEASURE_CALC_CNDTN_EXPR
Record 1	3	TOT_CALLS	1	Null	Null
Record 2	4	TOT_MOU	1	Null	Null

Associate a Measure with a Reporting Variable

Measure is a type of numeric variable that has some aggregation defined on it. This aggregation is defined in the BPP_RPT_MEASURE_MASTER table. However, you have to define the association of a measure with a variable in the BPP_RPT_MSR_X_VRBL table.

Table 13.15 *BPP_RPT_MSR_X_VRBL Table*

Column	Description
MEASURE_ID	the unique identifier for a measure that is used in workflow reports.
VARIABLE_ID	the unique identifier for a variable that is used in the workflow reports.

Example: Configure BPP_RPT_MSR_X_VRBL Table

You have defined the measures and their properties. To define a measure's association with a variable, add the following data into theBPP_RPT_MSR_X_VRBL table.

Table 13.16 *Sample Data in BPP_RPT_MSR_X_VRBL Table*

Column	MEASURE_ID	VARIABLE_ID
Record 1	23	22
Record 2	4	3

Configure a Derived Measure

Derived measures are calculated based on the simple measures. In the BPP_RPT_DRVD_MEASURE_MASTER table, you have to define the derived measures that you want to use for workflow reports.

Table 13.17 BPP_RPT_DRVD_MEASURE_MASTER Table

Column	Description
DRVD_MEASURE_ID	the unique identifier that is assigned to a variable that is used as a derived measure for workflow reports. The value for this column must be numeric.
DRVD_MEASURE_NM	the name of the derived measure.
AGGR_TYPE_ID	the unique identifier for type of aggregation that is performed on a measure. The list of aggregations is defined in the BPP_RPT_AGGR_MASTER table.
DRVD_MEASURE_MUL_FCT	is the multiplication factor for calculating the true value of a derived measure. This value is used in case of percentage or value conversion. The default value for this column is 1.
DRVD_MEASURE_OPRTR	the operator that is used in the definition of a derived measure. Currently, only the division operator is supported. Therefore, use the front slash (/) symbol to define the division operator.

Example: Configure BPP_RPT_DRVD_MEASURE_MASTER Table

You have defined the derived measures in the BPP_RPT_VRBL_MASTER table. Now, you have to define their properties in the BPP_RPT_DRVD_MEASURE_MASTER table. Add the following data into the BPP_RPT_DRVD_MEASURE_MASTER table.

Table 13.18 Sample Data in BPP_RPT_DRVD_MEASURE_MASTER Table

Column	Value
DRVD_MEASURE_ID	8
DRVD_MEASURE_NM	AVG_MOU_PER_CALL
AGGR_TYPE_ID	5
DRVD_MEASURE_OPRTR	/
DRVD_MEASURE_MUL_FCTR	1

There are certain limitations for configuring a derived measure:

- You have to define a derived measure based on two simple measures.
- You can configure a derived measure by using the division operation. You have to define the measures that will be used as the numerator and the denominator in the BPP_RPT_MSR_X_DRVD_MSR table.

- You cannot define a derived measure by using another derived measure.

Associate a Derived Variable with a Simple Measure

In order to define an association between a derived variable and a simple variable, enter appropriate values in the BPP_RPT_MSR_X_DRVD_MSR table.

Table 13.19 *BPP_RPT_MSR_X_DRVD_MSR Table*

Column	Description
DRVD_MEASURE_ID	the unique identifier that is assigned to a derived measure.
MEASURE_ID	the unique identifier for a variable that is defined as a measure for workflow reports.
MEASURE_NMRTR_IND	indicates whether the measure is used in the numerator part of the derived measure definition.
VARIABLE_ID	the unique identifier for a variable that is used in the workflow reports.

Example: Configure BPP_RPT_MSR_X_DRVD_MSR Table

You can define the derived measure, AVG_MOU_PER_CALL, which will be created by using the measures TOT_CALLS and TOT_MOU. To do so, add the following data into the BPP_RPT_MSR_X_DRVD_MSR table.

Table 13.20 *Sample Data in BPP_RPT_MSR_X_DRVD_MSR Table*

Column	DRVD_MEASURE_ID	MEASURE_ID	VARIABLE_ID	MEASURE_NMRTR_IND
Record 1	8	4	23	Y
Record 2	8	3	22	N

Associate a Report Category with a Reporting Variable

You have to associate the report category with the report variable. This association is a many-to-many relationship. In other words, a reporting variable can be available for multiple categories and vice versa. This association is defined in the BPP_RPT_CTGRY_X_VRBL table.

Table 13.21 *BPP_RPT_CTGRY_X_VRBL Table*

Column	Description
RPT_CTGRY_ID	the unique identifier of a report category.

Column	Description
VARIABLE_ID	the unique identifier for a variable that is used in the application.
VRBL_MNDTRY_IND	indicates whether it is mandatory to include the variable in workflow reports that are defined for the report category.
WORKFLOW_STEP_ID	the unique identifier of a workflow step.

Example: Configure BPP_RPT_CTGRY_X_VRBL Table

In order to ensure that all the measures and the derived measures that you have defined earlier are displayed in the SAS Offer Optimization for Communication interface, add the following data into the BPP_RPT_CTGRY_X_VRBL table.

Table 13.22 *Sample Data in BPP_RPT_CTGRY_X_VRBL Table*

Column	RPT_CTGRY_ID	VARIABLE_ID	WORKFLOW_STEP_ID	VRBL_MNDTRY_IND
Record 1	2	44	3	N
Record 2	2	45	3	N
Record 3	2	64	3	N

Prepackaged Data for Workflow Reports

Overview

SAS Offer Optimization for Communications provides prepopulated data for workflow reports. This data includes information about dimensions, measures, sources of measures, and derived measures.

Dimensions

The following table lists the prepopulated dimensions that users can use for defining workflow reports.

Table 13.23 *Prepopulated Dimensions*

Dimension Name	Description
BUSINESS_GROUP_NM	Business group name
TARGET_SEGMENT_NM	Target segment name

Dimension Name	Description
MCSGMT_BUSS_NM	Microsegment name
BASE_OFFER_PYMNT_MODE_NM	Payment mode name of base offer
CITY_NM	City name
STATE_NM	State name
COUNTY_NM	County name
REGION_NM	Region name
COUNTRY_NM	Country name
CHURN_BAND_NM	Churn band name
CAL_MONTH_NM	Calendar month name
PROFITABILITY_BAND_NM	Profitability band name
AGE_BAND_NM	Age band name
CAL_QUARTER_NM	Calendar quarter name
BASE_OFFER_NM	Base offer name
TENURE_ON_NETWORK_IN_DAYS	Tenure on network in days
TENURE_ON_BASE_OFFER_IN_DAYS	Tenure on base offer in days
TENURE_ON_OFFER_BUNDLE_IN_DAYS	Tenure on offer bundle in days

Measures

The following table lists the prepopulated measures that users can use for defining workflow reports. Measures are further categorized based on their purpose.

Table 13.24 *Prepopulated Measures*

Measure Name	Description
Group name: Customer Counts	
TOT_CUST_CNT_IN_TS_FLTR	Total customer count in the target segment
TOT_CUST_CNT_IN_MS_FLTR	Total customer count in the microsegment
Group name: Usage	

Measure Name	Description
TOT_CALLS	Total number of calls
TOT_MOU	Total minutes of usage
Data	
TOT_DATA_CALLS	Total data of event calls
TOT_DATA_VOLUME	Total data of event volume
Group name: Message	
TOT_MESSAGE	Total number of messages
TOT_MESSAGE_VOLUME	Total volume of all messages
Group name: Revenue	
TOT_GROSS_REVENUE	Total gross revenue
TOT_ARPU	Total ARPU
TOT_DATA_USG_ARPU	Total ARPU for data usage
TOT_MESSAGE_USG_ARPU	Total ARPU for message usage
TOT_VOICE_USG_ARPU	Total ARPU for voice usage
TOT_VOICE_CALL_REVENUE	Total revenue of voice calls
TOT_DATA_CALL_REVENUE	Total revenue of data calls
TOT_MESSAGE_CALL_REVENUE	Total revenue of messages
TOT_NUM_OF_RECHARGES	Total number of recharges
TOT_AMT_OF_RECHARGES	Total amount of recharges

Derived Measures

The following table lists the prepopulated derived measures that users can use for defining workflow reports. Derived measures are further categorized based on their purpose.

Table 13.25 *Prepopulated Derived Measures*

Derived Measure Name	Description

Derived Measure Name	Description
Group name: Customer Counts	
PCT_CUST_IN_MS	Percentage of customers in microsegments
Group name: Usage	
PCT_VOICE_CALL_REV_TO_TOT_REV	Percentage of voice call revenue to total revenue
AVG_MOU_PER_CUST_IN_TS	Average minutes of usage per customer in the target segment
AVG_MOU_PER_CUST_IN_MS	Average minutes of usage per customer in microsegments
AVG_MOU_PER_CALL	Average minutes of usage per call
Group name: Data	
PCT_DATA_CALL_REV_TO_TOT_REV	Percentage of data call revenue to total revenue
AVG_MEGABYTE_PER_CUST_IN_TS	Average data volume per customer in the target segment
AVG_MEGABYTE_PER_CUST_IN_MS	Average data volume per customer in microsegments
AVG_MEGABYTE_PER_SESSION	Average data volume per session
Group name: Message	
PCT_MSG_CALL_REV_TO_TOT_REV	Percentage of message call revenue to total revenue
AVG_MESSAGES_PER_CUST_IN_TS	Average number of messages per customer in the target segment
AVG_MESSAGES_PER_CUST_IN_MS	Average number of messages per customer in the micro segment

Chapter 14

Batch Processing

Processing a Project in Batch Mode

When you run all the workflow steps of a project in design mode, the project is ready to be promoted to batch mode. Using the **Push a project to batch mode** feature of SAS Offer Optimization for Communications, users promote the project from design mode to batch mode. The project IDs of projects, which are promoted to batch mode are stored in the BPP_PROJECT_SCHEDULE table along with the schedule frequencies.

To run a project in batch mode:

1. From the BPP_PROJECT_SCHEDULE table, identify the project whose schedule is due for the batch run.
2. Run the bpp_executeoocbatch.sas macro. This macro is located in the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASFoundForComm5.2/SASMacro** folder. Alternatively, you can schedule this macro in the SAS Enterprise Guide environment.
3. Specify the ID of the project as the input to the bpp_project_id macro parameter.
4. View the log file that is created in the **<SAS configuration path>/Lev1/Applications/SASOfferOptforComm5.2/Logs/ooc_logs/oocprojectlogs** folder. In this folder, a separate log file is created for each project. For example, for a project whose ID is 20, the log will be stored in the bpp_executeoocbatch_20.log file.

Note: There can be multiple projects that are due for the batch run at the same time. In this case, you can run all these projects in parallel. To do so, repeat steps 2 and 3.

Purging Batch Run Data

You might want to configure a workflow step of a project that is in batch mode. However, in order to do so, you have to first pull the project into design mode. You can pull a project into design mode by using the **Pull project to design mode** feature of the

SAS Offer Optimization for Communications interface. Before you use this feature, make sure that you purge all batch run data for that project.

To purge the batch run data for a project:

1. Go to the **<SAS configuration path>/Lev1/SASApp/SASEnvironment/SASFoundForComm5.2/SASMacro** folder.
2. Run the bpp_projectrun_purge_script.sas macro. Alternatively, you can also schedule this macro in the SAS Enterprise Guide environment.
3. Provide the project run ID as a value for the bpp_project_run_id parameter.

Part 3

Appendixes

Appendix 1

Data Management

Business Rules for Data Management

Data Model Assumptions

The SAS Offer Optimization for Communications data model is based on the following assumptions:

- The following tables are of Type I:
  - CUST_AGRMNT_D
  - PRODUCT_D
  - SERVICE_D
  - BUSINESS_GROUP_D
  - EQUIPMENT_D
  - EVENT_TYPE_D

  - CALENDAR_D
  - ANALYTICAL_MODEL_D
  - BILL_AGRMNT_D
  - BILL_CYCLE_D
  - BUSINESS_GEOGRAPHY_D
  - BUSINESS_GROUP_D
  - CALENDAR_D
  - CHARACTERISTICS_D
  - CHARGE_TYPE_D
  - CSP_USAGE_BUCKET_D
  - CUST_AGRMNT_D
  - DISCOUNT_D
  - ELIGIBILITY_ATTRBT_D
  - ELIGIBILITY_COMBINATION_D
  - EQUIPMENT_D
  - EVENT_TYPE_D
  - LOCATION_D
  - OFFER_BUNDLE_D
  - OFFER_D
  - PAYMENT_METHOD_D
  - PREPAID_SCHEME_D
  - PRODUCT_D
  - PROMOTION_D
  - RECHARGE_VOUCHER_D
  - SERVICE_D
  - TARIFF_D
  - TIME_BAND_D
- The CAL_SK column in the Calendar dimension should be a sequential number. The value for this column should increment by 1. Therefore, the higher the CAL_DT value, the higher will be the corresponding CAL_SK value. For example, if 01-Jan-2009 has an SK of 1020, then 02-Jan-2009 should have an SK of 1021.
- A customer can be either a prepaid or a postpaid customer. In other words, a customer can own an offer bundle that has the offer payment mode code as prepaid or postpaid. However, there can be scenarios in which communications service providers support one customer to own both prepaid and postpaid offer bundles. For these scenarios, the staging should be designed in order to assign a different customer ID for prepaid and postpaid offer bundles.
- A customer can own multiple offers. However, a customer can own only one offer bundle at a time.
- Only one model can be in production for a customer. SAS Offer Optimization for Communications does not handle champion-challenger types of scenarios.

- SAS Offer Optimization for Communications can be implemented for any currency. However, for a certain implementation, only one currency can be supported.
- The billing cycles end-day number represents the day of every month on which the rating system processes all the unbilled usage or non-usage charges in order to generate bills. Therefore, for a billing cycle that has an end day number of 10, the accumulated data from the 11th of last month until the 10th of the current month is considered. As a result, the following dates cannot be valid end day numbers for any billing cycle: 28, 29, 30, and 31 of any month.

ETL Assumptions

If certain scenarios have more than one source system for the same subject area, then staging should be designed to generate unique IDs for the Dimension tables.

The measures in the fact table of the Foundation data mart are populated with default values. In the implementation phase, the Foundation data mart ETLs should not set a NULL value for any of the measures.

Assumptions for Business Reports

The percentage variance of churned customer count is calculated based on the previous churned customer count. The reports display information based on this calculation. The formula for computing the percentage variance is as follows:

((Current period churned customers - Previous period churned customers) / Previous period churned customers) * 100

Scheduling ETL Jobs

The tables that are listed below provide information about the ETL flows and the dependencies that exist between various jobs of a group. This information will help you to schedule the ETL jobs. However, consider the following instructions before you schedule ETL jobs.

- Consider the order in which the jobs of a particular job group are to be run. The Group order column represents the parent group, and the Level within group column indicates the sublevels within each group.
- Execute the jobs in one group only after all jobs of previous groups are executed. For example, the jobs in group 2 should be started only when all the jobs in group 1 are executed.
- Within a group, jobs should be run based on their Level within group. For example, there can be two jobs within a group that have level number 1 and level number 2. In this case, Job 2 should be run only after Job 1 is complete.
- Within a group, jobs that have the same level number can be run in parallel.
- The bill monthly, monthly, and weekly jobs can be run in parallel.

Load Order Sequence for Bill Monthly Jobs

Table A1.1 *Load Order for Bill Monthly Jobs*

Group Order	Level Within Group	Job Name
1	1	cfd_param_bill_cycle_dt_job
1	2	cfd_param_bill_cycle_sk_job
1	3	cfd_param_bill_cycle_cd_job
1	4	cfd_param_bill_cyc_cd_sk_job
1	5	cfd_param_declaration_job
2	1	cfd_pst_pd_cus_x_bill_cycle_bridge_tmp_job
3	1	cfd_pst_pd_bill_usage_b_job
3	1	cfd_pst_pd_cust_acct_snpsht_b_job
3	1	cfd_pst_pd_cust_bill_b_job
3	1	cfd_pst_pd_cust_bill_nonusage_b_job
3	1	cfd_pst_pd_cust_interaction_b_job
3	1	cfd_pst_pd_cust_loyalty_b_job
3	1	cfd_pst_pd_equip_activity_b_job
3	1	cfd_pst_pd_payment_base_b_job
3	1	cfd_pst_pd_payment_drvd_b_job
3	1	cfd_pst_pd_service_activity_b_job
3	1	cfd_pst_pd_subscrp_bill_nonusage_b_job
3	1	cfd_pst_pd_subscrp_interaction_b_job
3	1	cfd_pst_pd_subscrp_loyalty_b_job
3	1	cfd_pst_pd_usage_b_job
3	1	cfd_pst_pd_cust_offer_snpsht_b_job
3	1	cfd_pst_pd_cust_snpsht_b_job
3	1	cfd_cust_bill_monthly_d_tmp_job

Group Order	Level Within Group	Job Name
3	2	cfd_cust_bill_monthly_f_tmp_job
3	2	cfd_cust_service_bill_mth_f_tmp_job
3	3	cfd_cust_prev_bill_mth_arpu_tmp_job
3	3	cfd_cust_service_prev_bill_mth_arpu_tmp_job
3	4	cfd_cust_bill_monthly_summary_incr_f_job
3	4	cfd_cust_service_bill_mth_summary_incr_f_job
3	5	cfd_cust_bill_monthly_summary_f_job
3	5	cfd_cust_service_bill_mth_summary_f_job
3	6	cfd_cust_bill_mth_summary_cube

Load Order Sequence for Monthly Jobs

Table A1.2 *Load Order for Monthly Jobs*

Group Order	Level within Group	Job Name
1	1	cfd_param_monthly_dt_update_job
1	1	cfd_param_monthly_sk_update_job
1	4	cfd_param_declaration_job
2	1	cfd_cust_monthly_d_tmp_job
2	2	cfd_cust_monthly_f_tmp_job
2	2	cfd_cust_service_mth_f_tmp_job
2	3	cfd_cust_service_prev_mth_arpu_tmp_job
2	3	cfd_cust_prev_mth_arpu_tmp_job
2	4	cfd_cust_service_monthly_summary_incr_f_job
2	4	cfd_cust_monthly_summary_incr_f_job
2	5	cfd_cust_monthly_summary_f_job
2	5	cfd_cust_service_mth_summary_f_job

Group Order	Level within Group	Job Name
2	6	cfd_cust_cmprtv_monthly_summary_f_job
2	6	cfd_cust_mth_bg_mvmnt_analysis_f_job
2	6	cfd_cust_mth_sgmnt_mvmnt_analysis_f_job
2	6	cfd_cust_monthly_csus_job
2	6	cfd_subscrp_monthly_csus_job
3	1	cfd_cust_mth_summary_cube
3	1	cfd_cust_mth_sgmt_mvmnt_cube
3	1	cfd_cust_mth_bg_mvmnt_cube
3	1	cfd_cust_cmprtv_mth_summary_cube
3	1	cfd_cust_offer_mth_csus_cube
3	1	cfd_cust_service_mth_csus_cube
3	1	cfd_subscrp_offer_mth_csus_cube
3	1	cfd_subscrp_srvc_mth_csus_cube
4	1	bpp_cust_rf_key_tmp_mth_job
4	2	bpp_cust_summary_rf_tmp_job
4	3	bpp_cust_mthly_summary_rf_job
5	1	bpp_cust_srvc_rf_key_tmp_mth_job
5	2	bpp_cust_summary_rf_tmp_job
5	3	bpp_cust_srvc_mthly_summary_rf_job

Load Order Sequence for Weekly Jobs

Table A1.3 *Load Order for Weekly Jobs*

Group Order	Level within Group	Job Name
1	1	cfd_param_weekly_dt_update_job
1	1	cfd_param_weekly_sk_update_job

Group Order	Level within Group	Job Name
1	2	cfd_param_declaration_job
2	1	cfd_pre_pd_cust_tmp_job
3	1	cfd_pre_pd_cust_interaction_b_job
3	1	cfd_pre_pd_cust_loyalty_b_job
3	1	cfd_pre_pd_equip_activity_b_job
3	1	cfd_pre_pd_service_activity_b_job
3	1	cfd_pre_pd_subscrp_bucket_drvd_b_job
3	1	cfd_pre_pd_subscrp_bucket_usage_b_job
3	1	cfd_pre_pd_subscrp_interaction_b_job
3	1	cfd_pre_pd_subscrp_loyalty_b_job
3	1	cfd_pre_pd_subscrp_usage_drvd_b_job
3	1	cfd_pre_pd_usage_b_job
3	1	cfd_pre_pd_usage_recharge_b_job
3	1	cfd_pre_pd_cust_offer_snpsht_b_job
3	1	cfd_pre_pd_cust_snpsht_b_job
3	1	cfd_cust_weekly_d_tmp_job
3	2	cfd_cust_weekly_f_tmp_job
3	2	cfd_cust_service_weekly_f_tmp_job
3	3	cfd_cust_service_prev_weekly_arpu_tmp_job
3	3	cfd_cust_prev_weekly_arpu_tmp_job
3	4	cfd_cust_service_weekly_summary_incr_f_job
3	4	cfd_cust_weekly_summary_incr_f_job
3	5	cfd_cust_service_weekly_summary_f_job
3	5	cfd_cust_weekly_summary_f_job
3	6	cfd_cust_cmprtv_weekly_summary_f_job
3	6	cfd_cust_weekly_csus_job

Group Order	Level within Group	Job Name
3	6	cfd_subscrp_weekly_csus_job
3	7	cfd_cust_weekly_summary_cube
3	7	cfd_cust_service_wkly_csus_cube
3	7	cfd_cust_offer_weekly_csus_cube
3	7	cfd_subscrp_offer_wkly_csus_cube
3	7	cfd_subscrp_srvc_wkly_csus_cube
4	1	bpp_cust_rf_key_tmp_weekly_job
4	2	bpp_cust_summary_rf_tmp_job
4	3	bpp_cust_weekly_summary_rf_job
5	1	bpp_cust_srvc_rf_key_tmp_mth_job
5	2	bpp_cust_summary_rf_tmp_job
5	3	bpp_cust_srvc_wk_summary_rf_job

Customizing the Foundation Data Mart

Overview

Initially, the scope of the Foundation data mart can be wider than the scope of the data that is to be subsequently loaded. You might customize the Foundation data mart in order to align it with the changing business requirements.

Considering time consistency is an important factor for customizing the Foundation data mart. It is important for the application that is accessing the Foundation data mart that the new data is loaded to the same historical point as the current Foundation data mart.

Customizing the Foundation data mart can include the following requirements:

- Update the data source system extract files.
- Update the SAS Data Integration Studio jobs.
- Change the dependencies, Foundation data mart tables, and indices.
- Add new jobs in SAS Data Integration Studio for downstream applications.

Before you begin customizing the Foundation data mart, make sure that you have complete information about the data flow from the source systems to the Foundation data mart. First identify each component that has an impact and then modify each component accordingly in the test environment.

Make sure that you identify the updates that you make with a suitable prefix. This will ensure that a column or a table is not duplicated in the new version of the Foundation data mart. Also, make sure that all the updates have meaningful descriptions.

Examples

The following requirements can lead to customizing the Foundation data mart:

Add a new table
: A new table can be added in order to support new subject areas that are not covered in the Foundation data mart. You can distinguish the new table by adding a prefix such as X to the table name.

Add a new column
: A new column can be added to an existing table in order to store data that is not captured by the Foundation data mart. You can distinguish the new column by adding a prefix such as X to the column name.

Increase field lengths
: You can increase the field length if data from the upstream systems is not fully accommodated in the fields with current lengths. However, you must consider the impact of increasing field length and also ensure that the change in the field length does not result in truncating data from downstream systems.

Change formats and informats
: You can change the formats and informats in order to address the local language requirements.

Store descriptions in different languages
: Reference tables enable you to store descriptions in multiple languages. Use the LANGUAGE_CD column in the reference table to specify the language for the description. Consider the downstream impact on the ETL process if a description is stored in more than one language.

Categories

You can customize the Foundation data mart at three levels:

Simple changes
: These changes do not affect the data structure, logic, or any process in the data warehouse. For example, changes such as increasing the length of character columns, updating the labels or formats of certain columns, or adding descriptive columns that are not referenced by supplied applications can be simple changes.

Moderate changes
: These changes are localized and clear and have a simple effect on the logic or process. For example, changes such as adding data columns to certain tables can be a moderate change.

Complex changes
: These changes have a widespread effect with potential complex impact on logic or process. For example, adding a new table can be a complex change.

The degree of complexity is based on how much manual effort is required and how many components are affected. Therefore, it is recommended, that you avoid making complex changes whenever possible because they add to the cost and make future upgrades more difficult.

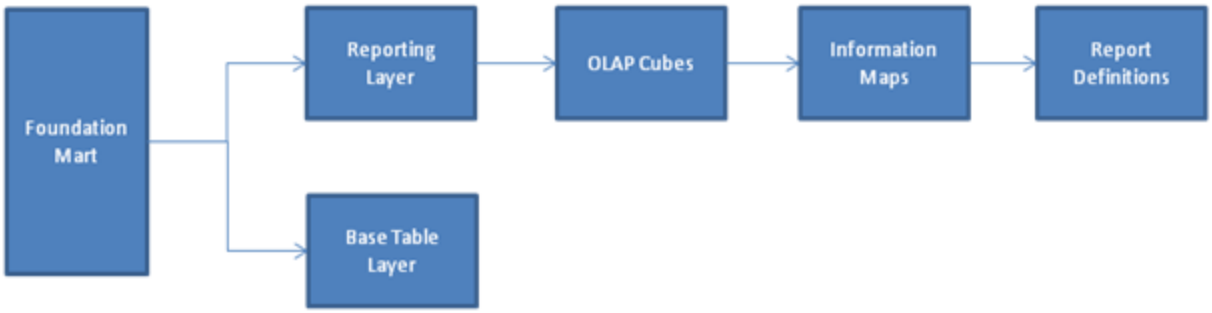
Scope

You can customize the following components:

- Foundation data mart data model
- reporting data model and corresponding ETL
- base layer data model
- information maps
- report definitions
- data management processes

The following figure indicates the cascading effect of the changes that are made to these components.

Figure A1.1 *Customization Flow*



Note: This document focuses on instructions for customizing the Foundation data mart.

Steps

To customize the Foundation data mart:

1. Identify the requirements.
   a. Identify the gaps between requirements and the solution's predefined data mart.
   b. For each additional variable that is required in the target data mart, determine whether the required data is available in the Foundation data mart.

      Note: In certain cases, the gaps might not be apparent. If aggregated or derived variables are available in source systems or existing data warehouses, the variables might be directly added to the Foundation data mart tables. This can help to avoid business rules or aggregations being duplicated in solutions.
2. Map the requirements to identify the Foundation data mart customization. Based on the Foundation data mart gap analysis, identify the specific additions to the Foundation data mart.
   a. Identify the columns to add to the existing tables.
   b. In case the gaps between requirements and the solution are not fulfilled by adding new columns, identify additional tables that need to be added to fill this gap. Also, identify relationships of the new tables with the existing tables.
3. Implement the identified customization.

Note: Any changes that are made to the predefined Foundation data mart environment should be logged. Details of this log are mentioned in the subsequent section.

Modifying the Foundation Data Mart

The following modifications are required while implementing a Foundation data mart at a customer's site:

- Add local language or value format requirements.
- Add columns in the tables for business requirements.
- Add an entity or a table for a new area of business.
- Add new data formats.
- Customize the language code (LANGUAGE_CD) to be specific to the country.
- Customize most of the amount values that use currencies (CURRENCY_CD) to be specific to the country.
- Change column labels (which can be done with no impact).
- Expand the length of the columns when data is expected to be longer. This will not have any impact on the existing data but processes that refer to this particular column will need to be updated.

Impact

Before customizing the Foundation data mart, consider the impact that it might have on the underlying applications and processes.

- Columns can be added to tables. However, if these new columns are used in downstream applications that have already been implemented, customize the downstream applications. While adding a column, use a prefix (for example, X_) to indicate that it is a user-added column.
- The length of the columns cannot be reduced. This change might impact downstream processing.
- Tables cannot be added. However, if these new tables are used in downstream applications that have already been implemented, customize the downstream applications. If a new business area is added, new processes should be added instead of modifying existing processes. While adding a table, use a prefix (for example, X_) to indicate that it is a user-added table.

Restrictions

Avoid the following modifications while customizing the Foundation data mart:

- Do not change column names. This change can significantly impact downstream applications that have already been implemented.
- Do not change data types for columns. This change can significantly impact downstream applications that have already been implemented.
- Do not delete columns. Columns that are not required by the client can hold missing values.

- Do not reduce the length of columns. This change can significantly impact downstream processing.

Important Tips

While customizing the Foundation data mart, ensure that the following prerequisite tasks are also performed:

- Develop and test modifications to the Foundation data mart in a controlled development or test environment before the modifications are migrated to a production environment.
- Tables or columns can be added or the length of a column can be modified within the SAS Data Integration Studio environment. See the SAS Data Integration Studio User's Guide for more information.
- If the supplied DDL files are changed, back up or copy the original DDL files to revert to a known starting point.
- Identify the job dependencies and job order for new jobs and schedule the jobs accordingly.
- While creating user-defined tables and columns, ensure that they contain meaningful and accurate descriptions. To provide meaningful and accurate descriptions, all descriptions should be clear and concise without circular reasoning.
- If the historical ARPU values are already available, then communications service providers can use these values for further calculating the ARPU values, by loading them into CUST_PREV_MTH_ARPU_TMP, CUST_SERVICE_PREV_MTH_ARPU_TMP, CUST_PREV_WEEKLY_ARPU_TMP, CUST_PREV_BILL _MTH_ARPU_TMP, and CUST_PREV_WEE KLY_ARPU_TMP tables at initial load.

Appendix 2

Parameter Configuration

Parameters for DABT

In order to populate the configuration area and ABT-specific area of the Analytics data mart, you have to run certain macros and configure certain parameters of these macros. The following table lists the macro parameters. Make sure that you configure the parameters that are marked as editable.

Note: If you configure values for parameters that are non-editable, then the code can produce erroneous results.

Table A2.1 *CFDN_CONFIG_PARAM Table*

Parameter Name	Description	Is Editable	Value
flx_src_lib	required by DABT internal code. This parameter contains the location of the Analytics data mart table.	Yes	dabt_adm
flx_tmp_lib	required by DABT internal code. This parameter stores the location of the intermediate tables.	Yes	scrflx
cfdn_err_lib	required by the DABT interface code. This parameter stores the location in which the error data structures reside.	Yes	err_lib
cfdn_code_subsetquery	required by the wrapper code in order to identify source tables, which would be used as the source for subset query.	No	SQY

Parameter Name	Description	Is Editable	Value
cfdn_code_measures	required in the wrapper code. This parameter determines the code that is to be used to indicate an attribute of measure type. It is used when accepting source table column-related inputs from the user from the worksheet.	No	MSR
cfdn_code_dimensions	required in the wrapper code. This parameter determines the code that is to be used to indicate an attribute of dimension type. It is used when accepting source table column-related inputs from the user from the worksheet.	No	DIM
cfdn_code_dates	required in the wrapper code. This parameter determines the code that is to be used to indicate an attribute of date type. It is used when accepting ABT variable related inputs from the user from the worksheet.	No	DAT
code_for_keys	required in the wrapper code. This parameter determines code that is to be used to indicate an attribute of KEY type. It is used when accepting ABT variable related inputs from the user from the worksheet.	No	KEY
cfdn_code_timeprds	required in the wrapper code. This parameter determines the code that is to be used to indicate an attribute of datetime type. It is used when accepting ABT variable related inputs from the user from the worksheet.	No	TMP

Parameter Name	Description	Is Editable	Value
cfdn_code_aggregations	required in the wrapper code. This parameter determines the code that is to be used to indicate an attribute of aggregation type. It is used when accepting ABT variable related inputs from the user from the worksheet.	No	AGG
cfdn_code_suppl	required in the wrapper code. This parameter indicates the code that is to be used for a supplementary variable.	No	SPM
cfdn_code_recent	required in the wrapper code. This parameter indicates the code that is to be used for a recent variable.	No	RNT
cfdn_code_behavioral	required in the wrapper code. This parameter indicates the code that is to be used for a behavioral variable.	No	BEH
cfdn_code_derived	required in the wrapper code. This parameter indicates the code that is to be used for a derived variable.	No	DER
cfdn_code_invalid	required in the wrapper code. This parameter indicates the code that is to be used for columns for which usage is not specified in the configuration data.	No	INV
cfdn_modeling_abt_suffix	required in the wrapper code. This parameter indicates the suffix that is to be used for a modeling ABT.	No	_mod
cfdn_scoring_abt_suffix	required in the wrapper code. This parameter indicates the suffix that is to be used for a scoring ABT.	No	_scr

Parameter Name	Description	Is Editable	Value
cfdn_code_create	iused by wrapper codes. This parameter indicates the code that is to be used for invoking the generate list macro in creation mode as well as creating ABT.	No	C
cfdn_code_save	used by wrapper codes. This parameter indicates the code that is to be used for invoking the generate list macro in saving mode.	No	S
cfdn_code_delete	used by wrapper codes. This parameter indicates the code that is used for deleting an ABT.	No	D
cfdn_code_accept	used by wrapper codes. This parameter indicates the code that is used for accepting a change in the edited ABT variable list.	No	A
cfdn_code_update	used by wrapper codes. This parameter indicates the code that is used for updating a row in the edited ABT variable list.	No	U
cfdn_code_origexists	used by DABT. This parameter indicates whether the variable that is to be created already exists in the variable_master table.	No	O
cfdn_code_rename	used by DABT. This parameter indicates whether the variable that is to be created is renamed.	No	H
cfdn_code_template	used by the wrapper codes. This parameter indicates the code for template entry for the ABT from which modeling and scoring ABT entries are copied.	No	T
cfdn_code_prepaid_microseg	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates that the purpose of the ABT is “microsegmentation ABT for prepaid.”	Yes	MSPRE

Parameter Name	Description	Is Editable	Value
cfdn_code_postpaid_microseg	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates that the purpose of the ABT is “microsegmentation ABT for postpaid.”	Yes	MSPOST
cfdn_code_prepaid_seg	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates that the purpose of the ABT is “segmentation ABT for prepaid.”	Yes	CSPRE
cfdn_code_prepaid_churn	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates that the purpose of the ABT is “churn ABT for prepaid.”	Yes	CRPRE
cfdn_code_postpaid_seg	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates that the purpose of the ABT is “segmentation ABT for postpaid.”	Yes	CSPOST
cfdn_code_postpaid_churn	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates that the purpose of the ABT is “churn ABT for postpaid.”	Yes	CRPOST
cfdn_code_prepaid_offerrank	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates that the purpose of the ABT is “offer ranking ABT for prepaid.”	Yes	ORPRE

Parameter Name	Description	Is Editable	Value
cfdn_code_postpaid_offerrank	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the purpose of the ABT is "offer ranking ABT for postpaid."	Yes	ORPOST
cfdn_template_abt_library	used for internal comparisons in wrapper and integration codes. This parameter is also used in configuration area tables.	No	TMP
cfdn_source_library	used for internal comparisons in wrapper and integration codes. This parameter is also used in configuration area tables.	No	SOU
cfdn_abt_library	used for internal comparisons in wrapper and integration codes. This parameter is also used in configuration area tables.	No	ABT
cfdn_libsfile_nm	used in churn and segmentation internal codes for writing the library declarations that are to be used in SAS Enterprise Miner.	Yes	libs_include.sas
cfdn_sp_rc	used in error macros and also places where error macros are called.	No	0
cfdn_err_threshold	used in error macros and also places where error macros are called.	No	4
cfdn_code_modeling	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the code for modeling ABT.	No	M
cfdn_code_scoring	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the code for scoring ABT.	No	S

Parameter Name	Description	Is Editable	Value
cfdn_flag_on	used for internal comparisons in wrapper codes.	No	Y
cfdn_flag_off	used for internal comparisons in wrapper codes.	No	N
cfdn_commasep	the character that represents a comma in wrapper codes.	No	,
cfdn_hashsep	the character that represents a hash in wrapper codes.	No	#
cfdn_uscrsep	the character that represents an underscore in wrapper codes.	No	_
cfdn_tmprdsep	used as a code for indicating time period separator in the internally generated variable description.	Yes	IN
cfdn_dimsep	used as code for indicating a dimension values separator in the internally generated variable description.	Yes	FOR
cfdn_aggsep	used as code for indicating an aggregations separator in the internally generated variable description.	Yes	OF
cfdn_colsep	the character that represents a colon in wrapper codes.	No	:
cfdn_quotesep	the character that represents quotation marks in wrapper codes.	No	%str(%')
cfdn_recentsep	the code to indicate a recent value separator in the internally generated variable description.	Yes	last
cfdn_semicolsep	the character that represents a semicolon in wrapper codes.	No	;

Parameter Name	Description	Is Editable	Value
cfdn_code_last	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the beginning of a recent variable that is the last value.	No	LST_
cfdn_code_day_freq	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the daily frequency.	No	DAY
code_init	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the code for initialization.	No	I
code_month	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the monthly frequency.	No	MONTH
cfdn_code_select_dt	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the date value for select.	No	SELECT
cfdn_code_recent_dt	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the date value for recent.	No	RECENT
cfdn_code_extraction_dt	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the date value for extraction.	No	DT_EXTR

Parameter Name	Description	Is Editable	Value
cfdn_code_validto_dt	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the date value for valid to.	No	VALID_TO
cfdn_code_validfr_dt	used in the integration codes of SAS Offer Optimization for Communications. This parameter indicates the date value for valid from.	No	VALID_FR
cfdn_none_join_type	used by wrapper codes. This parameter indicates the code for the NONE value of the JOIN_TYPE column (SUBSET_TABLE_JOIN_CONDITION table). The NONE value indicates that only one table is participating in the join and therefore, the joining condition does not exist.	No	NONE
cfdn_code_lenissue	used by wrapper codes in variable edit code. This parameter indicates the value for length issues with variables.	Yes	VAR_NM_GT_28
cfdn_code_noissue	used by wrapper codes in variable edit code. This parameter indicates the value for no issues with variables.	Yes	OK
cfdn_code_null	used by wrapper codes. This parameter indicates the null value and is used for internal comparisons.	No	'
cfdn_code_na	used by wrapper code. This parameter indicates an invalid value.	No	NA
cfdn_abt_wrksht	the worksheet name for ABT.	Yes	abt
cfdn_project_wrksht	the worksheet name for a project.	Yes	project
cfdn_suppl_wrksht	the worksheet name for a supplementary variable.	Yes	Supplementary

Parameter Name	Description	Is Editable	Value
cfdn_beh_wrksht	is the worksheet name for behavioral variable.	Yes	beh
cfdn_recent_wrksht	the worksheet name for a recent variable.	Yes	Recent
cfdn_drvdsrc_wrksht	the worksheet name for derived variable source.	Yes	derivedvarsource
cfdn_drvddts_wrksht	the worksheet name for derived variable details.	Yes	derivedvardetails
cfdn_subsetwhere_wrksht	the worksheet name for a WHERE condition of a subset query.	Yes	subset_where
cfdn_library_wrksht	the worksheet name for a library.	Yes	Library
cfdn_tmfreq_wrksht	the worksheet name for a time frequency.	Yes	Time_Frequency
cfdn_aggtype_wrksht	the worksheet name for an aggregation.	Yes	Aggregation_Type
cfdn_purpose_wrksht	the worksheet name for a purpose.	Yes	Purpose
cfdn_level_wrksht	the worksheet name for a level.	Yes	Levels
cfdn_tmprd_wrksht	the worksheet name for a time period.	Yes	Time_Period
cfdn_srctabusg_wrksht	the worksheet name for source table usage.	Yes	Source_Table_Usage
cfdn_srctab_wrksht	the worksheet name for a source table.	Yes	Source_Table
cfdn_srccolusg_wrksht	the worksheet name for source column usage.	Yes	Source_Column_Usage
cfdn_srccoltyp_wrksht	the worksheet name for a source column type.	Yes	Source_Column_Type
cfdn_srccol_wrksht	the worksheet name for a source column.	Yes	Source_Columns
cfdn_dimcolvals_wrksht	the worksheet name for ABT dimensional attribute values.	Yes	Dim_Attribute_Values

Parameter Name	Description	Is Editable	Value
cfdn_vartype_wrksht	the worksheet name for a variable type.	Yes	Variable_Type
cfdn_sbst_frmpth_wrksht	the worksheet name for a from path.	Yes	Subset_From_Path
cfdn_frmpth_lev_wrksht	the worksheet name for from path level.	Yes	From_Path_Level
cfdn_sbstjoin_wrksht	the worksheet name for a join condition for subset.	Yes	Subset_Join_Condition
cfdn_code_null	the code for null. This parameter is used for comparisons in the internal code.	No	'
cfgxls	the path for storing the DABT configuration (configuration area) Excel worksheet.	Yes	c:\bpp\common\dabt \OR_BPPConfiguration_08 032010.xls
abtxls	the path for storing ABT definition (ABT-specific area) Excel worksheet.	Yes	c:\bpp\common\dabt \OR_ABT_SPECIFIC.xls
cfdn_tmpabt_config_ind		No	N
cfdn_num_vartype_id	used for writing the template ABT details to the BPP_VRBL_MASTER table as a one-time activity.	No	1
cfdn_num_vartype_cd	used for writing the template ABT details to the BPP_VRBL_MASTER master as a one-time activity.	No	NUM
cfdn_cat_vartype_id	used for writing the template ABT details to the BPP_VRBL_MASTER table as a one-time activity.	No	2
cfdn_cat_vartype_cd	used for writing the template ABT details to the BPP_VRBL_MASTER table as a one-time activity.	No	CAT
cfdn_bpp_rollback_flg	indicates whether to support rollback in SAS Offer Optimization for Communications.	Yes	Y

Parameter Name	Description	Is Editable	Value
cfdn_nonbpp_rollback_flg	indicates whether to support rollback in Churn and Segmentation modules.	Yes	Y
cfdn_postpaid_time_freq	used in the integration code of SAS Offer Optimization for Communications while creating offer ranking ABT.	No	month
cfdn_sk_value	only a global declaration. This parameter does not require a value.	No	
cfdn_dim_val_list	only a global declaration. This parameter does not require a value.	No	
cfdn_dim_list	only a global declaration. This parameter does not require a value.	No	
cfdn_msr_list	only a global declaration. This parameter does not require a value.	No	
cfdn_dat_list	only a global declaration. This parameter does not require a value.	No	
ord_cfdn_dat_list	only a global declaration. This parameter does not require a value.	No	
cfdn_tmprd_list	only a global declaration. This parameter does not require a value.	No	
cfdn_agg_list	only a global declaration. This parameter does not require a value.	No	
cfdn_drvd_var_abt_var_list	only global declaration. This parameter does not require a value.	No	
log_flx	only a global declaration. This parameter does not require a value.	No	
cfdn_var_sk_list	only a global declaration. This parameter does not require a value.	No	

Parameter Name	Description	Is Editable	Value
cfdn_var_nm_list	only a global declaration. This parameter does not require a value.	No	
cfdn_var_display_nm_list	only a global declaration. This parameter does not require a value.	No	
cfdn_var_desc_list	only a global declaration. This parameter does not require a value.	No	
cfdn_path_log	only a global declaration. This parameter does not require a value.	No	
cfdn_sp_rc_err_msg	only a global declaration. This parameter does not require a value.	No	
cfdn_code_rename	only a global declaration. This parameter does not require a value.	No	
cfdn_code_noissue	only a global declaration. This parameter does not require a value.	No	
cfdn_empty_dimval	only a global declaration. This parameter does not require a value.	No	
cfdn_empty_msr	only a global declaration. This parameter does not require a value.	No	
cfdn_empty_dim	only a global declaration. This parameter does not require a value.	No	
cfdn_empty_tmprd	only a global declaration. This parameter does not require a value.	No	
cfdn_empty_agg	only a global declaration. This parameter does not require a value.	No	
ca_smd	the Library in which the smd data set resides. The ca_smd library internally points to sashelp.	No	ca_smd

Parameter Name	Description	Is Editable	Value
ca_dabt_smd_error	the name of the locale-specific smd data set, which has messages to be displayed in the log.	No	ca_dabt_smd_error
lib_abt	the name of the library in which the Dynamic ABT data mart resides.	No	dabt_adm

Parameters for Customer Analytics

You have to configure certain parameters before you build analytical models for customer retention, customer segmentation, and cross-sell and up-sell.

Table A2.2 *Parameters for Customer Retention and Customer Segmentation*

Parameter Name	Value	Data Type	Description	Is Editable
ca_smd	ca_smd	C	the name of the library in which the smd data set resides. The ca_smd library internally points to sashelp.	N
ca_crcs_smd_error	ca_crcs_smd_error	C	the name of the locale-specific smd data set, which has messages to be displayed in the log.	N
cfdn_src_sys_cd	S_1	C	the code that is given to the source system. This indicates the system (for example, billing and CRM) from which data is populated in the Foundation data mart.	Y
cfdn_abt_folder_nm	abt	C	the name of the folder in which the DABT project-specific ABT would reside.	N
cfdn_art_folder_nm	art	C	the name of the folder in which the DABT project-specific ART would reside.	N

Parameter Name	Value	Data Type	Description	Is Editable
cfdn_scrcd_folder_nm	scrcd	C	the name of the folder in which the DABT project-specific SAS Enterprise Miner score code would reside.	N
cfdn_model_folder_nm	model	C	the name of the folder in which the SAS Enterprise Miner project workspace for a certain ABT and DABT project would reside.	N
cfdn_reten_soln_abbr_nm	cr	C	the abbreviation to indicate “churn” as the purpose of the ABT that is being built.	N
cfdn_segm_soln_abbr_nm	cs	C	the abbreviation to indicate segmentation as the purpose of the ABT that is being built.	N
cfdn_apdm_lib	dabt_adm	C	the name of the library, which refers to the Analytics data mart.	N
cfdn_abt_var_assoc_tbl_nm	abt_x_variable	C	the name of the table from the Analytics data mart, which indicates the association between an ABT and the variables.	N
flg_on	Y	C	the value of the flag that satisfies certain conditions.	N
cfdn_varbl_master_tbl_nm	variable_master	C	the name of the Variable Master table from the Analytics data mart.	N
cfdn_cust_scr_dtl_tbl_nm	cust_model_score_dtl	C	the name of the table from the Foundation data mart that stores customer-level model scores.	N

Parameter Name	Value	Data Type	Description	Is Editable
cfdn_subscrp_scr_dtl_tbl_nm	subscrp_model_score_dtl	C	the name of the table from the Foundation data mart that stores subscription-level model scores.	N
cfdn_tmp_lib_nm	scratch	C	the name of the library that stores temporary tables. These tables are created during the scoring phase for a DABT project.	N
cfdn_tmp_abt_abbr	_TMP	C	the suffix that is appended to all the ABT names, which are of temporary type.	N
cfdfact	fact	C	the name of the library, which refers to the FACT schema of the Foundation data mart.	N
cfdn_subsdimnm	subscrp_d	C	the name of the subscription table from the Foundation data mart that stores subscription-level attributes.	N
cfdn_custdimnm	cust_d	C	the name of the customer table from the Foundation data mart that stores customer-level attributes.	N
cfdn_art_lib_nm	art	C	the name of the library, which refers to the ART folder of the DABT project workspace.	N
cfdn_aggr_churn_scrs_flg	Y	C	the flag that indicates whether the subscription- level churn scores are to be aggregated to customer level.	N

Parameter Name	Value	Data Type	Description	Is Editable
cfdn_cust_sgmt_dtl_tbl_nm	cust_analytical_sgmt_dtl	C	the name of the table of the Foundation data mart that stores customer-level segment codes.	N
cfdn_tmp_folder_nm	scratch	C	the name of the folder in which all the temporary tables, which are output of the scoring activity, would reside.	N
cfdn_error_folder_nm	error	C	the name of the folder in which all the error tables would reside. These tables contain error messages that are (if any) produced for every step of the ABT processing codes.	N
cfdn_log_folder_nm	log	C	the name of the folder in which all logs of the ABT processing macros would reside.	N
startposvar	1	N	an integer that specifies the position at which the search should start and the direction of the search. This variable is used to extract the modeling ABT name from the scoring project name.	N
cfdn_sp_rc	0	N	used to check the error code status in common foundation macros.	N
cfdn_error_threshold	4	N	the maximum value of sp_rc or sqlrc variables for which the SAS process encountered a situation for which it issued a warning. The statement continued to execute.	N

Parameter Name	Value	Data Type	Description	Is Editable
cfdn_err_tbl_lib_nm	error	C	the name of the library, which refers to the location where all the error tables would reside. These tables contain error messages (if any) that are produced for every step of the ABT processing codes	N
log_clear	new	C	indicates whether the SAS statements are executed to append the existing log or create a new log file.	N
cfdn_conf_lib	CFDMISC	C	the name of the library, which refers to the MISC schema and stores all the parameter or configuration data sets. These data sets are required for building churn and segmentation ABTs.	Y
cfdn_segm_model_typ_cd	SEGM	C	indicates that the model is of type customer segmentation.	N
cfdn_xsus_model_typ_cd	XSUS	C	indicates that the model is of type cross-sell and up-sell.	N
cfdn_churn_model_typ_cd	CHURN	C	indicates that the model is of type customer retention.	N
cfdn_cust_xsus_scr_tbl_nm	CUST_MODEL_STEPLEVEL_SCORE_DTL	C	contains the name of the customer-level Foundation data mart table that stores the scores at a level other than customer.	N
cfdn_sub_xsus_scr_tbl_nm	SUBSCRP_SCORE_STEPLEVEL_DTL	C	contains the name of the Foundation data mart subscription-level table that stores the scores at a level other than subscription.	N

Parameter Name	Value	Data Type	Description	Is Editable
cfdn_pred_model_run_tbl_nm	PREDICTIVE_MODEL_RUN_DTL	C	contains the name of the Foundation data mart table that stores details about scoring runs for a predictive model.	N
cfdn_srvcdim	SERVICE_D	C	contains the name of the Foundation data mart service table that stores service-level attributes.	N
cfdn_aggr_xsus_scrs_flg	Y	C	flag that indicates whether the subscription-level cross-sell AND up-sell scores are to be aggregated to the customer level.	Y
cfdn_xsus_level_co_nm	CUSTOFFR	C	contains the name of the customer-offer level.	N
cfdn_xsus_level_ss_nm	SUBSCRPSRV	C	contains the name of the subscription-service level	N
cfdn_segm_model_run_tbl_nm	SEGMENTATION_MODEL_RUN_DTL	C	contains the name of the Foundation data mart table that stores details about scoring runs for a segmentation model.	N
cfdn_ofrdim	OFFER_D	C	contains the name of the offer table of the Foundation data mart that stores offer- level attributes.	N
cfdn_entity_level_service	SERVICE	C	indicates whether the entity level is service.	N
cfdn_entity_level_offer	OFFER	C	indicates whether the entity level is offer.	N
cfdn_xsus_soln_abbr_nm	XU	C	contains the abbreviation that indicates that the purpose of the ABT is cross-sell and up-sell.	N

Parameter Name	Value	Data Type	Description	Is Editable
cfdn_anmod_dtl_tbl_nm	ANALYTICAL_MODEL_DTL	C	contains the name of the Foundation data mart table that stores the association between the analytical model and entity for which the model is being built.	N

Appendix 3

Derived Variables

The **deriveddetails** worksheet of the ABT-specific workbook defines a derived variable. The definition includes the type of the derived variable as simple or complex. In addition, it also defines the expression, which is the logic based on which the value of the derived variable is to be computed.

The **derivedsource** worksheet of the ABT-specific workbook lists the source for the derived variable that is being defined. The source variables must belong to the ABT for which the variables are being defined. The source variables are mostly the ABT variables that appear in the expression from the **deriveddetails** worksheet.

Example: Simple Derived Variable

For example, the offer ranking ABT expects that certain variables that are to be derived must be present in the ABT. To populate these variables in the **deriveddetails** worksheet, you have to define the name and description of the variable along with the name of the ABT that the variable belongs to. As this variable derives its value from an ABT variable and not from another derived variable, the value of the derived_var_flg must be set to S. The source variables for this variable are: SUM_BLR_TBLA_B1M and SUM_BLR_TDSA_B1M. The TOTAL_BILL_AMT_B1M (Total Bill amount) derived variable is calculated by subtracting the billing amount from the discount amount. The expression is written accordingly.

Table A3.1 *Example of Simple Derived Variable*

ABT_Nm	Derived_Var_Nm	Derived_Var_short_Nm	Derived_Var_Desc	Derived_Var_Flg	Expression
BPP_ORPOST_CUST	TOTAL_BILL_AMT_B1M	TOTAL_BILL_AMT_B1M	TOTAL_BILL_AMT_B1M	S	SUM_BLR_TBLA_B1M - SUM_BLR_TDSA_B1M

Table A3.2 *Simple Derived Variable and ABT Details*

ABT_Nm	Derived_Var_Nm	ABT_Var_Nm
BPP_ORPOST_CUST	TOTAL_BILL_AMT_B1M	SUM_BLR_TBLA_B1M
BPP_ORPOST_CUST	TOTAL_BILL_AMT_B1M	SUM_BLR_TDSA_B1M

Here, the assumption is that the SUM_BLR_TBLA_B1M and SUM_BLR_TDSA_B1M variables are defined as variables as a part of the BPP_ORPOST_CUST. ABT.

Example: Complex Derived Variable

You want to define a variable that calculates the proportion of peak downloaded data volume in the total downloaded data volume over the last three months. In order to calculate the value for this simple derived variable, you have to consider the following two ABT variables:

- SUM_PK_DNL_PSU_VL_L3M
- SUM_OPK_DNL_PSU_VL_L3M

Assume that the user-defined name of the derived variable is PK_BY_OP_PRPN_L3M. The value for PK_BY_OP_PRPN_L3M can be calculated by using the expression that is mentioned in the table below

Table A3.3 *Example of Simple Derived Variable*

ABT_Nm	Derived_Var_Nm	Derived_Var_short_Nm	Derived_Var_Desc	Derived_Var_Flg	Expression
BPP_CSPOST_CUST	PK_BY_OP_PRPN_L3M	PK_BY_OP_PRPN_L3M	PK_BY_OP_PRPN_L3M	S	SUM_PK_DNL_PSU_VL_L3M / (SUM_PK_DNL_PSU_VL_L3M + SUM_OPK_DNL_PSU_VL_L3M)

Table A3.4 *Simple Derived Variable and ABT Details*

ABT_Nm	Derived_Var_Nm	ABT_Var_Nm
BPP_CSPOST_CUST	PK_BY_OP_PRPN_L3M	SUM_PK_DNL_PSU_VL_L3M
BPP_CSPOST_CUST	PK_BY_OP_PRPN_L3M	SUM_PK_DNL_PSU_VL_L3M

Based on the simple derived variable, PK_BY_OPK_PRPN_L3M, you now want to define a complex derived variable that enables you to define a preference for the peak usage flag. In this case, you can derive the value of this flag by considering the value of the PK_BY_OPK_PRPN_L3M variable. If the value of PK_BY_OPK_PRPN_L3M is greater than 0.7, then this flag should have a value 1. Otherwise, the value of this flag will be 0.

Assume that the user-defined name of this complex derived variable is PREF_PK_USG.. You can compute the value for this variable by using the expression that is mentioned in the table below:

Table A3.5 *Example of Complex Derived Variable*

ABT_Nm	Derived_Var_Nm	Derived_Var_short_Nm	Derived_Var_Desc	Derived_Var_Flg	Expression
BPP_CSPOST_CUST	PREF_PK_USG	PREF_PK_USG	PREF_PK_USG	C	CASE WHEN (PK_BY_OPK_PRPN_L3M > 0.7) THEN 1 ELSE 0 END

Table A3.6 *Complex Derived Variable and ABT Details*

ABT_Nm	Derived_Var_Nm	ABT_Var_Nm
BPP_CSPOST_CUST	PREF_PK_USG	PK_BY_OP_PRPN_L3M

Glossary

ABT variable
a column in an analytical base table. ABT variables are used to build a statistical model to predict defaults.

analytical base table
a highly denormalized data structure that is designed to build an analytical model or to generate scores based on an analytical model. Short form: ABT.

average revenue per user
the average amount of revenue that a communications service provider collects from each subscriber each month. Communications service providers often try to boost this number by selling extra services to subscribers, such as data plans, messaging, and downloadable content.

business group
a subset of the customer base that is derived as a result of high-level business segmentation based on relatively static business attributes such as offer segment (wireless, land-line), offer payment mode (prepaid, postpaid), customer type, and customer's geographical area.

campaign
a marketing initiative with a number of components that are designed to achieve a commercial goal. A campaign can have one or many communications.

churn
the process of losing active customers and their related revenue. Churn can be classified as either voluntary or involuntary, depending on the reason for disconnecting the subscription. Note: Subscribers who have migrated to prepaid services are not considered churners and are tracked separately.

churn score
a process that uses analytical data and process models to predict the likelihood of customer churn. The churn models are developed based on data from account, client, household, subscription, and equipment information. The

churner
a subscriber that involuntarily or voluntarily disconnects a subscription.

communications service provider
a company or organization that provides communication services as a business. Communications service providers might operate networks, or they might integrate

the services of other providers in order to deliver end-to-end service to their customers. The term communications service provider is now being used generically and might include Telecom Service Providers (TSPs), Internet Service Providers (ISPs), Application Service Providers (ASPs) and other organizations that provide services.

cube
See OLAP cube.

dimension
a group of closely related hierarchies. Hierarchies within a dimension typically represent different groupings of information that pertains to a single concept. For example, a Time dimension might consist of two hierarchies: (!) Year, Month, and Date, and (2) Year, Week, and Day.

ETL
the process of (1) obtaining data from operational sources (the extract step), (2) cleansing and preparing data for import into the data warehouse (the transform step), and (3) importing the transformed data into the data warehouse (the load step).

fact
a single piece of factual information in a data table. For example, a fact can be an employee name, a customer's phone number, or a sales amount. It can also be a derived value such as the percentage by which total revenues increased or decreased from one year to the next.

hierarchical list
a user interface element that helps to select values by organizing variables into parent-child relationships, typically where a parent member represents the consolidation of its children. A hierarchical list progresses from top to bottom.

information map
a collection of data items and filters that provides a user-friendly view of a data source. When you use an information map to query data for business needs, you do not have to understand the structure of the underlying data source or know how to program in a query language.

offer
a base offer to which a customer is subscribed.

OLAP
See online analytical processing.

OLAP cube
a logical set of data that is organized and structured in a hierarchical, multidimensional arrangement to enable quick analysis of data. A cube includes measures, and it can have numerous dimensions and levels of data.

online analytical processing
a software technology that enables users to dynamically analyze data that is stored in multidimensional database tables (cubes).

SAS Display Manager
an interactive, windowing interface to SAS System software. SAS Display Manager commands can be typed on the command line, entered by pressing function keys, and selected from the PMENU facility.

SAS Enterprise Guide
a software application with a point-and-click interface that gives users access to the functionality of many components of SAS software. Interactive dialog boxes guide users through data analysis tasks and reporting tasks, and users can easily export the results of those tasks to other Windows applications or servers. SAS Enterprise Guide provides access not only to SAS data files, but also to data that is in a wide variety of other software vendors' formats and in other operating system formats.

SAS Metadata Server
a multi-user server that enables users to read metadata from or write metadata to one or more SAS Metadata Repositories.

star schema
tables in a database in which a single fact table is connected to multiple dimension tables. This is visually represented in a star pattern. SAS OLAP cubes can be created from a star schema.

workflow
the sequence of activities that is to be performed for each project.

workflow report
a report that you can define for each step of the project workflow.

workflow step
each individual activity of a project that is depicted in a workflow diagram.

Index

Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.