

THE ABC'S OF OAB APIs

Johnson Louis, BOSS Corporation
Patti Kingsmore, BOSS Corporation

INTRODUCTION

Implementing Oracle Advanced or Standard Benefits requires many unique design and configuration techniques. This application in the HRMS suite of products is a separate mini implementation in itself. Although integrated with Human Resources and Payroll, Benefits should not be underestimated. Even current Oracle users should consider this as a re-implementation. One option many clients consider is implementing Benefits in Phase 2 of an overall HRMS implementation.

Most organizations have a well-documented benefits program that they offer to their employees. Until 11i it had to be managed outside of the suite of applications. Organizations both new to Oracle and upgrading from a prior release have the opportunity to configure Oracle Benefits to meet many if not all of their benefit offerings to employees and related business rules in the integrated HRMS system.

Oracle Benefits offers extensive, configurable functionality within the base product. From eligibility profiles which determine which of your employees are eligible for which programs and plans within those programs to configurable self-service benefits that allow your employees to make elections during open enrollment and view their benefits via the internet throughout the year.

Along with this extensive functionality comes a large amount of design, configuration, setup, and conversion.

As with most implementations of Oracle Applications, conversions require both complex decisions and tedious mapping, loading, and verification of data. Benefits add new and unique challenges to the typical scenario. Once application setup is complete and a conversion approach is determined, data mapping involves structuring compensation object elections from either payroll elements or sometimes legacy codes.

Once the mapping is complete, the next daunting task is to methodically use api's (Application Program Interfaces) to place the "Orac-lized" data into the application. Hold your sigh of relief, once api processing is complete there are a few more steps you must endure. You must run the participation process to determine the eligibility, electability, rates and flex credits for the elections you mapped in.

This paper will discuss one approach to the conversion of legacy benefit information into Oracle Advanced Benefits. It will examine both the functional decisions and technical methods involved. The detailed technical focus will concentrate on the use of the APIs such as the sequencing, required inputs, methodology, etc.

BACKGROUND

Advanced Benefits (OAB) is an extension of Basic Benefits and contains much of the needed functionality for managing a full benefits offering in-house. Some of the main features found only in Advanced Benefits include:

- Life Event Management
- Flex Credit Management for cafeteria-style benefit plans
- Managing reimbursement requests for Flexible Spending Accounts
- Communication kits for open enrollment and life event processing
- What-if functionality to assist with enrollment elections

Advanced Benefits is a separate licensed module and must be purchased. When existing 10.7 and 11.x users purchase 11i they must also purchase this module to gain its additional functionality. You do not have to purchase Oracle Payroll to use Advanced Benefits. Therefore, Oracle users must make a very important decision when they implement their 11i strategy.

IMPLEMENTATION STRATEGIES

Current 11.x and 10.7 users have a huge upgrade task when they decide to implement 11i. The task is complex enough without having to completely redesign benefit management. Because Standard Benefits still functions under 11i, users should seriously consider postponing this benefit redesign until they have completed the migration to 11i. This will allow time to resolve all 11i issues before undertaking this complex project. It might also allow the new benefit functionality to mature, as there are some initial quality issues. There will also be more resources to draw from when you need assistance with your implementation. This includes resources that come cost-free such as list servers and user groups. Standard and Advanced Benefits offer many choices and decisions that should be studied carefully without the pressure of completing an 11i upgrade project.

Assume that you are now ready to begin your task of implementing benefits using the new functionality that Oracle offers. There are several implementation strategies that you can use to migrate your existing benefits to the new structures. Each one will be discussed in the following sections.

COMPLETE REINSTALL

Some existing 10.7 and 11.x users have decided to create a new install of the Oracle Applications Release 11i and use existing APIs to move their data into this new instance. This strategy will allow you to rebuild all your existing elements and structures using new benefits functionality.

CONVERSION VERSUS PROCESS NEW ELECTIONS

As discussed previously, the benefit models in 11i are significantly different from previous releases. There are really no APIs available that will take current benefit data in Oracle and automatically create your programs and plans in 11i. Therefore, you must rebuild your benefit programs with this new functionality. Once this process is complete, you still have some big decisions to make on how to get your current employee base into this new model.

Oracle delivers a series of APIs that will allow you to load current and historical elections directly into the benefit tables. This assumes that you have already built and tested your benefit programs in 11i. You must first decide if it is necessary to load any historical election information into 11i. This might be necessary if you want to default new elections based on historical elections. Certain enrollment requirements may also be driven from historical election. This is common in dental and vision plans, which may require a minimum enrollment period before changes are allowed. U.S. customers are now required to produce HIPAA certifications to prove coverage history. OSB/OAB would not be able to produce this information until at least 18 months of history has been accumulated in these modules.

Loading historical information can be complex and very time consuming. You must map your legacy data to the new Oracle definitions and write the necessary code to load this into the APIs. You may want to shorten this process by loading in only the current elections. Clearly, the advantages of having this information at your go-live date must be weighed against the time and effort it will take to load this information.

The second approach is to use open enrollment to load in current enrollment information. Every participant must re-enroll in the new benefit programs using either Self Service, or by completing new election forms to be entered by benefits personnel. This will allow participants to "clean up" their benefit elections, including dependent and beneficiary designations. This approach is simpler and should be considered for users who are on very tight implementation schedules. There is no need to perform any data mapping from the legacy data to the new benefit definitions.

The drawbacks to this second approach are many. No elections can be defaulted, because there is no current enrollment information in the system. You will not be able to enforce any election rules either, because OSB/OAB will not know the current enrollment information. As stated previously, HIPAA certifications will have to be completed outside of the system until enough history has been processed in OSB/OAB.

The task of implementing Advanced Benefits is a major component of your 11i upgrade or install. Make sure you explore all your options before making the best decision for your organization.

OVERVIEW OF OAB CONFIGURATION TOPICS

COMPENSATION OBJECTS

Compensation objects are the building blocks of benefits. Navigating down the Compensation Object hierarchy, programs are at the highest level, followed by plan types, plans, and options. Similar to other Oracle Applications, you build from the bottom up. Once the compensation objects are defined and you have completed some of the basic setup steps such as defining your program/plan years, you begin to expand on the configuration of your benefits offerings. Attach eligibility profiles for both participant and dependent coverage to limit which people are eligible for your offerings. You configure life event functionality to determine what changes to a person's HR record creates potential enrollment opportunities or changes to current enrollments both during open enrollment and during the plan year. You define communications to be sent to participants. You configure variable rate profiles and

standard rates that can be automatically sent to Oracle Payroll for processing. If your program offers flex credits such as a cafeteria style offering, define the amount of flex credits provided and how a participant can spend those flex credits.

PROGRAMS

Programs are the highest level in the compensation hierarchy and therefore the last compensation object configured. Programs group plans that have similar eligibility requirements and the same enrollment periods for participants to enroll in. It is the umbrella for your benefits offerings. The types of programs you can set up include Core, COBRA, and Flex Credit programs. At the program level, you set the currency, define the activity reference period, frequencies for enrollment and rates, and the periods your program covers. Also you define which plans and plan types are included in the program.

Define the participation start and end dates, any waiting period that exists for the program, and any maximum enrollment periods for each program. You also attach eligibility profiles or rules to determine which of your employees are eligible to participate. For example, your employees might be required to be full-time regular employees or part-time regular employees with at least 1000 hours worked to be eligible for the program.

PROGRAM ENROLLMENT REQUIREMENTS

For each program you define, you must also set up the enrollment requirements. The process of determining a person's electable choices is based on the requirements you define for programs and at the plan levels, if different. Although participation eligibility has already been defined for the program, it is the enrollment requirements that create electable choices by defining the rules associated with enrolling in a program or changing current elections.

You can define the enrollment method, enrollment code, coverage dates, and rate start and end dates. Determine whether or not the program allows unrestricted enrollment, default enrollments, and automatic enrollments. Also, configure default enrollment codes for those participants that fail to respond to an active life event.

You must also define the timing related to enrolling in a program. For open enrollment and for the conversion life events, define the start and end dates for enrollment and changes to current elections, when to close enrollment, and the coverage and rate start and end dates. For each life event that affects the program, define the period of enrollment or window of time the employee has to elect or make changes to his enrollment.

Finally, you can set the dependent coverage designations at the program or plan type level, any certifications such as social security number or date of birth needed for the dependent, dependent eligibility profiles, and changes you can make to dependent coverage based on a given life event.

PLAN TYPES

Plan types group similar plan offerings. Examples include medical, dental, flexible spending accounts, and life insurance. These plan classifications allow you to define the types of options a participant can elect and the minimum and/or maximum number of plans a participant can enroll in concurrently.

PLANS

Plans are the individual offerings available to an employee if they meet the enrollment requirements. Plans can be either in a program or a separate offering, termed plan not in program. Examples include a medical plan offered in a program or a savings plan that is independent of any program offering.

OPTIONS

Options represent the choices your employees have for coverages or elections within the plans they enroll in. For each option, define which plan types, and therefore plans, the option is available for. By defining Participant plus Family, an employee can elect to cover himself and his family under the medical plan or defining Six Units would allow an employee to select a level of employee life insurance coverage.

In addition to defining which plan types the options are available for, you can configure the designation requirements for each option. Based on an employee's contact records, for example, spouse and children, define the personal relationships that are required to select the contact during enrollment. In addition to specifying which relationship types are available, you also enter the minimum and maximum number required. If you do not define the designation requirements, there are no restrictions on who can be designated.

ELIGIBILITY PROFILES AND DERIVED FACTORS

Once the compensation objects have been defined and the life events functionality has been configured, you must control who can enroll in the benefits you offer. If there are no eligibility profiles attached to your program, everyone is eligible. By attaching an eligibility profile to a compensation object, you restrict who can enroll. The profiles can be attached to a program, plan type in program, plan in program, plan, and option in plan. If an employee is eligible for all plans in the program, simply define the eligibility requirements once at the program level because eligibility cascades down the compensation object hierarchy.

Eligibility profiles work with life events (Advanced Benefits) and enrollment requirements to create electable choices for the benefit package your enterprise offers. Eligibility is determined during the Participation Process for Advanced Benefit users.

Derived benefit eligibility factors offer a systematic way to establish when a person is eligible for benefit offerings. The criteria you can choose from includes compensation level, hours worked, percent full-time, age, length of service, and combined age and length of service. By defining the thresholds required for your requirements, the system can alert you to a potential eligible participant.

FLEX CREDITS AND IMPUTED INCOME CONSIDERATIONS

Many enterprises offer a cafeteria-style benefits program to their employees. By providing a calculated number of benefits flex credits to eligible employees, the employees are able to select from the available selections, the options and coverages to create their own personalized benefits package. The concept of creating a flex credit program includes calculating the credits available to each employee, how the employees can spend the flex credits, how excess and deficit credits are handled, the structure of the program including the plans and options offered to employees, and the configuration of the payroll elements to process the employees' elections during the payroll cycle.

Your benefit program may also require you to set up imputed income. If you offer coverages for an employee's domestic partner, you may need to do some extra setup and configuration upfront which will allow the system to detect, record, and report imputed income automatically for those benefits.

IMPLEMENTATION STEPS

DEFINE MISCELLANEOUS BASIC SETUP STEPS

- Add the Benefits Tabbed Region to the People Window
- Define a Monthly Benefit Payroll
- Define Regulations

SETUP ELIGIBILITY FACTORS AND PROFILES

- Define Benefits Groups
- Define Postal Code (ZIP) Ranges
- Define Service Areas
- Define Derived Factors
 - Compensation Level Factors
 - Percent Full Time Employment Factors
 - Hours Worked in Period Factors
 - Age Factors
 - Length of Service Factors
 - Combination Age and Length of Service Factors
- Define Participant Eligibility Profiles
- Define Dependent Coverage Eligibility Profiles

DEFINE LIFE EVENT PROCESSING

- Define Life Event Processing and Person Changes
- Define Related Person Changes and Associate with Life Events

Define Collapsing Life Events

DEFINE COMPENSATION OBJECTS AND ENROLLMENT REQUIREMENTS

Define Reimbursable Goods and Service Types
Define a Program or Plan Year Period
Define Plan Types
Define Options
Define Plans
Define Programs
Define Program and Plan Enrollment Requirements

DEFINE RATES, COVERAGES, PREMIUMS

Calculate Variable Activity Rates
Calculate Coverages
Define Across Plan Type Coverage Limits
Calculate Imputed Income
Calculate Actual Premium Costs
Define Period-to-Date Limits
Define Earning and Deduction Benefit Elements
Define Activity Rates for Standard Contributions
Define Flex Credits and Benefit Pools

CONVERSION CONFIGURATION STEPS

After the design and configuration of OAB and the implementation approach has been decided, the following steps must be performed in the application before conversion.

BENEFIT EXTRACT CHANGE EVENTS

The Extract Change Event Log creates an audit log of changes to a participant's elections, etc. These triggers can be used if your enterprise only reports changes to the benefit carriers. For conversion, these triggers need to be disabled. Navigate to the Application Utilities Lookups (HR Lookups) and query BEN_EXT_CHG_EVT. Uncheck the enabled box for each change trigger.

POTENTIAL LIFE EVENT CONFIGURATION OPTIONS

There are several approaches to control potential life events. Potential life events are triggered based on the person changes defined as part of the configuration of OAB. One is to wait to define person life events in the Production instance until after the conversion is complete.

If the life events are part of the “gold” copy of your Production instance, define the Data Migrator Mode. This pay action parameter suppresses the person life event triggers defined in OAB when HR data is converted. Add the `DATA_MIGRATOR_MODE` and set the value to “P” on the Pay Action Parameters form.

If neither of the above approaches were used, you can do one of two things: 1) Set the conversion life events to Overriding or 2) Create a simple SQL script to void all existing potential life events prior to running the conversion API's.

SETUP API LIFE EVENTS

LIFE EVENTS

To convert Oracle Advanced Benefits using API's you must define 2 life events as a part of the functional setup.

- **Current Enrollment** – used in the conversion process and the data from the legacy system is loaded using APIs.
- **Conversion** – used in the post conversion process when the Participation Process (Concurrent Manager Program) is run. Participation Process uses the recently loaded data and creates the actual Benefit Enrollments.

PROGRAM ENROLLMENT REQUIREMENTS

The program enrollment requirements defined for these life events are critical to the success of your conversion. The configuration depends on both the business requirements of your enterprise and the timing of your conversion / go-live dates.

Some important items to note:

CLOSE ENROLLMENT DATE TO USE

Since these life events are used only for conversion purposes, set the Close Enrollment Date to “when enrollment period ends.” When running the Close Enrollment Process, this code will help ensure default election processing is assigned.

ENROLLMENT PERIOD START / END DATES

Again, since these life events are used only for conversion, set the enrollment period start and end dates to “As of Event Date.”

DEFAULTS

Define the program enrollment default processing for participants. In our scenario, we used plan type level defaults on the program enrollment requirements to determine how the participation process should process the elections converted via API's. For the Conversion Life Event, set the Default Enrollment Code for each plan type to “New Defaults, Current, Same Enrollment but Default Rates.” The exception in our scenario was to set the code to “...Same Enrollment and Rates” for the Flexible Spending Accounts. The amounts mapped in from legacy via the Current Life Event would be defaulted to the participant's enrollment.

Also Days After Enrollment Period to Apply Defaults should be defined on the timing tab of the program enrollment requirements. This ensures the system assigns defaults when the Close Enrollment process is run.

COVERAGE / RATE START / END DATE

Program enrollment requirements also determine when coverage and rates start and end for each life event. The codes of enrollment coverage start date/end date and rate start date/end date for the API life events should coincide with the dates used for all other defined life events. It is very important to use codes that do not create overlapping element dates or gaps in coverage or rate dates. Examples include start as of event and end 1 day before event or start first of pay period on or after event and end at the end of pay period on or after event.

DESIGNATION REQUIREMENTS

For the API life events, it is important to define dependent coverage program enrollment requirements. Based on your enterprise's requirement, define whether Program Dependent Designation is Optional or Required. Determine the Required Dependent Information, which could include a social security number, date of birth, address, and / or certifications such as birth certificate.

Add any Dependent Eligibility Profiles required for your program. Make sure you add the API life events to the Dependent Change of Life Event Options form. This ensures the api's actually attach designees to your plans. For example, set the code to "May Add Dependents."

RECONFIGURE SETUP FOR CONVERSION

The process of loading all of your pre-existing benefit information may be a multiple step process. There may be new functionality available in Oracle that wasn't available in the legacy system or some data in the legacy system that isn't "clean." One option is to have information required in Oracle but determine not to convert it. Beneficiaries are a great example. You may have the designation requirements set to require beneficiaries but determine not to convert the information from the legacy system. Before conversion, you can set required to optional, perform the conversion and then reset the enrollment requirement. You could then have your employees use Self Service Benefits to not only verify their benefit elections and dependents that were converted but to also enter the beneficiaries for those plans that require them.

IDENTIFYING AND MAPPING OF LEGACY DATA

The technical side of OAB conversion is all about bringing data from the legacy system into Oracle Advanced Benefits system. One of the major tasks is identifying the data within the legacy schema that is going to be ported into OAB. Not all the data may be required for conversion.

A custom table is required in the Oracle database to store the legacy data to make the conversion easy and fast. The custom table is designed according to the required legacy data structure. Once the table is created, a SQL script has to be developed to load the legacy data into the custom staging table. There are a variety of ways that one can adopt to load the legacy data into the custom staging table.

- Data can be read directly from the legacy system and loaded into staging
- ASCII data file can be opened & read by SQL and/or PL/SQL and loaded

- SQL*Loader can be used to load the data from ASCII files

The translation and mapping is another key area, which requires attention and accuracy. Translation is converting the legacy codes and names into its corresponding OAB terminology.

E.g.

Legacy Code	Translation
OC1	Open Choice Aetna
S	Individual Plus Spouse

Mapping is like solving a scrambled Rubik's Cube. It re-arranges the relationship in the right way within the entity based on the translation to create a clear road map and makes the conversion easy. Mapping includes the legacy data mapping with OAB and also mapping of OAB Plans & Plan Options with Elements & Tax Type.

E.g.

Plan Name	ADD-Individual
Plan Option Name	One Unit
Element Name	ADD Post Tax
Tax Type	After Tax

ABOUT ORACLE API

Oracle Advanced Benefit APIs (Application Programming Interface) has been provided to ease the data conversion process. APIs are written using Oracle's PL/SQL, an English like language with conditional statements, iterations, arithmetic, character, and date functions etc.

The APIs can be used as an alternative entry point into the system instead of the traditional online forms. The advantage of using a publicly callable API is that the API is using the same logic as the seeded online forms thus ensuring the data integrity in the database. Also, the APIs are fully supported and protect the user from any structural changes to the schema.

When calling an API you only need to specify the required parameters, which are those that do not have defaults, and those parameters that you wish to enter information against.

Oracle Applications APIs have a salient feature that allows using them in Validate mode without the data being written in the database. When an API is used in Validate mode "TRUE", the inputs are validated, a new record is created just like a real scenario record without a Commit. Also, it brings out the information about the record being

created falsely. Any error during the validation process will be detected and reported, followed by a complete Rollback.

The Object Version Number (OVN) controls record locking within the application. When a new row is inserted into the database, the OVN is usually set to 1. When it is subsequently updated, it is incremented by 1. Whenever a row is read and transferred to a client, the OVN is always transferred and this is used to determine if the record has been changed since the user retrieved the original data. This is the reason why the OVN is so important when calling APIs. When creating new records, the parameter is defined as an OUT parameter and is always initialized. During an update the parameter is In/Out and must be passed IN.

Some of the API have an Effective date parameter, which is a datetrack parameter, and is the date when the particular insertion or change becomes effective. It will only be present in those APIs, which are performing operations on datetracked tables.

Also, the APIs never issue a Commit. It is the responsibility of the calling program to take care of the Commits.

CONVERSION STEPS

To create the legacy data in the newly configured OAB system, the OAB APIs have to be called in a sequence. This is based on what data is going to be converted in the new OAB system. The scenario, which is being discussed here, is based on the approach that converts the legacy benefit data without any history and COBRA information.

SEQUENCING STEPS

- Step 1: Create Potential Life Event for Current LE
- Step 2: Update Potential Life Event
- Step 3: Create Life Event
- Step 4: Create Participant Enrollments
- Step 5: Create Elements & Rates
- Step 6: Create Eligible Covered Dependents
- Step 7: Create Eligible Beneficiaries
- Step 8: Close Life Event
- Step 9: Create Potential Life Event for Conversion LE

METHODOLOGY

The method involved in this conversion process is named as “Step Method”. This methodology requires a log table in Oracle database. The log table should have the capability of storing the key details of the data that is being processed by the API.

Also, a wrapper program is required to be designed and developed to call the APIs in the sequence. The program is responsible to read the legacy data, mapping data, and do translation before it passes the required values to the API. It is also responsible to write the log data in the log table, issue commit, etc.

When the wrapper program is executed a record is read from the staging table and the key values are recorded in the log table. Then the required parameters are validated and passed into the required API. When the API completes, the result values will be updated in the log table. This iteration will go on until all the steps are done successfully or stopped unsuccessfully midway. Once all the steps are complete for a record or whenever a step is stopped because of an error, the iteration is stopped and the next available record will be taken for processing. When a record completes with success the record in the legacy and in the log will be updated with a Success flag along with the Step number. In the event of an error, the log will be updated with a Failure flag and with the Step number along with the error message. The error record has to be corrected before the wrapper is executed next time.

To speed up the process, commit is issued after processing certain number of records.

STEP 1: CREATE POTENTIAL LIFE EVENT FOR CURRENT LIFE EVENT

The API used is `ben_ptnl_ler_for_per_api.create_ptnl_ler_for_per`

REQUIRED VALUES

- Life Event Occurred Date (`p_lf_evt_ocrd_dt`)
- Potential Life Event Reason for Person Status Code (`p_ptnl_ler_for_per_stat_cd`)
- Enrollment Period (`p_enrt_perd_id`)
- Life Event Reason (`p_ler_id`)
- Person Id (`p_person_id`)
- Business Group Id (`p_business_group_id`)
- Unprocessed Date (`p_unprocd_dt`)
- Effective Date (`p_effective_date`)

OUTVALUES

- Potential Life Event Reason for Person Id(`p_ptnl_ler_for_per_id`)
- Object Version Number(`p_object_version_number`)

RESULT

A new Potential Life Event record with a status of “Unprocessed” is created in the database.

STEP 2: UPDATE POTENTIAL LIFE EVENT

The API used is `ben_ptnl_ler_for_per_api.create_ptnl_ler_for_per`

REQUIRED VALUES

- Life Event Occurred Date (`p_lf_evt_ocrd_dt`)
- Potential Life Event Reason for Person Status Code (`p_ptnl_ler_for_per_stat_cd`)
- Enrollment Period Id (`p_enrt_perd_id`)
- Life Event Reason (`p_ler_id`)
- Person Id (`p_person_id`)
- Business Group Id (`p_business_group_id`)
- Detected Date (`p_dtctd_dt`)
- Processed Date (`p_procd_dt`)
- Notification Date (`p_ntfn_dt`)
- Effective Date (`p_effective_date`)

OUTVALUES

- Potential Life Event Reason for Person Id (`p_ptnl_ler_for_per_id`)
- Object Version Number (`p_object_version_number`)

RESULT

This API updates the Potential Life Event record for an employee and sets its status “Processed”.

STEP 3: CREATE LIFE EVENT FOR AN EMPLOYEE

The API used is `ben_person_life_event_api.create_person_life_event`

REQUIRED VALUES

- Person in Life Event Reason Status Code (`p_per_in_ler_stat_cd`)
- Life Event Occurred Date, Processed Date (`p_lf_evt_ocrd_dt`)
- Notification Date (`p_ntfn_dt`)
- Potential Life Event Reason for Person (`p_ptnl_ler_for_per_id`)
- Life Event Reason (`p_ler_id`)
- Person Id (`p_person_id`)
- Business Group Id (`p_business_group_id`)

OUTVALUES

- Person In Life Event Reason (p_per_in_ler_id)
- Process Date(p_procd_dt)
- Start Date(p_strtd_dt)
- Voided Date(p_voidd_dt)
- Object Version Number(p_object_version_number)

RESULT

A new Life Event is created

STEP 4: CREATE PARTICIPANT ENROLLMENTS

The API used is ben_prtt_enrt_result_api.create_prtt_enrt_result

This is the API that loads all current election information into OAB database. Most of the translation and mapping has to be done in this step.

REQUIRED VALUES

- Business Group Id (p_business_group_id)
- Option In Plan Id(p_oipl_id)
- Person Id (p_person_id)
- Program Id (p_pgm_id)
- Plan Id (p_pl_id)
- Plan Type in Program Id (p_ptip_id)
- Plan Type Id(p_pl_typ_id)
- Life Event Reason Id(p_ler_id)
- Unit of Measurement(p_uom)
- Original Enrollment Date(p_orgnl_enrt_dt)
- Enrollment Method Code(p_enrt_mthd_cd)
- Enrollment Coverage Start Date(p_enrt_cvg_strt_dt)
- Enrollment Coverage Through Date(p_enrt_cvg_thru_dt)
- Person in Life Event Reason Id(p_per_in_ler_id)
- Plan Order Number(p_pl_ordr_num)

- Plan in Program Order Number(p_plip_ordr_num)
- Option in Plan Order Number (p_oipl_ordr_num)
- Compensation Level Code(p_comp_lvl_cd)

OUTVALUES

- Participant Enrollment Result Id (p_prtt_enrt_rslt_id)
- Effective Start Date (p_effective_start_date)
- Effective End Date (p_effective_end_date)
- Object Version Number(p_object_version_number)

RESULT

Enrollment results are created.

STEP 5: CREATE ELEMENTS & RATES

The API used is ben_prtt_rt_val_api.create_prtt_rt_val

REQUIRED VALUES

- Person Id (p_person_id)
- Input Value Id (p_input_value_id)
- Element Type Id (p_element_type_id)
- Rate Start Date (p_rt_strt_dt)
- Rate End Date (p_rt_end_dt)
- Tax Type Code (p_tx_typ_cd)
- Activity Type Code (p_acty_typ_cd)
- Mlt Code (p_mlt_cd)
- Activity Reference Period Code (p_acty_ref_perd_cd)
- Rate Value (p_rt_val)
- Annual Rate Value (p_ann_rt_val)
- Communicated Rate Value (p_cmcd_rt_val)
- Communicated Reference Period Code (p_cmcd_ref_perd_cd)
- Elections Made Date (p_elctns_made_dt)
- Participant Enrollment Result Id (p_prtt_enrt_rslt_id)
- Person in Life Event Reason Id (p_per_in_ler_id)
- Activity Base Rate Id (p_acty_base_rt_id)
- Business Group Id (p_business_group_id)
- Effective Date (p_effective_date)

OUTVALUES

- Participant Rate Value Id (p_prtt_rt_val_id)
- Object Version Number (p_object_version_number)

RESULT

Elements and Rates are created.

STEP 6: CREATE ELIGIBLE COVERED DEPENDENTS

The API used is `ben_elig_cvrd_dpnt_api.create_elig_cvrd_dpnt`

This API creates the eligible covered dependents details in the OAB database.

REQUIRED VALUES

- Business Group Id (p_business_group_id)
- Participant Enrollment Result Id (p_prtt_enrt_rslt_id)
- Dependent Person Id (p_dpnt_person_id)
- Coverage Pending Flag (p_cvg_pndg_flag)
- Overridden Flag (p_ovrdn_flag)
- Person in Life Event Reason Id (p_per_in_ler_id)
- Effective Date (p_effective_date)

OUTVALUES

- Eligible Covered Dependent Id (p_elig_cvrd_dpnt_id)
- Effective Start Date (p_effective_start_date)
- Effective End Date (p_effective_end_date)
- Object Version Number(p_object_version_number)

RESULT

Eligible covered dependents are created.

STEP 7: CREATE ELIGIBLE BENEFICIARIES

The API used is `ben_plan_beneficiary_api.create_plan_beneficiary`

Eligible beneficiaries are created in this step.

REQUIRED VALUES

- Business Group Id (p_business_group_id)
- Participant Enrollment Result Id (p_prtt_enrt_rslt_id)
- Benefit Person Id (p_bnf_person_id)
- Primary Contingent Code (p_prmry_cntngnt_cd)
- Person in Life Event Reason Id (p_per_in_ler_id)
- Effective Date (p_effective_date)

OUTVALUES

- Plan Beneficiary id (p_pl_bnf_id)

- Effective Start Date (p_effective_start_date)
- Effective End Date (p_effective_end_date)
- Object Version Number(p_object_version_number)

RESULT

Beneficiaries' records are created.

STEP 8: CLOSE LIFE EVENT

The API used is ben_person_life_event_api.update_person_life_event

This is the last step in the enrollment of Current Enrollment life event. This API closes the enrollment and sets the life event status as "Processed".

REQUIRED VALUES

- Person in Life Event Reason Id (p_per_in_ler_id)
- Person in Life Event Reason Status Code (_per_in_ler_stat_cd)
- Life Event Occurred Date (_lf_evt_ocrd_dt)
- Notification Date (p_ntfn_dt)
- Potential Life Event Reason for Person Id(_ptnl_ler_for_per_id)
- Life Event Reason Id (p_ler_id)
- Person Id (p_person_id)
- Business Group Id (p_business_group_id)
- Effective Date (p_effective_date)

OUTVALUES

- Process Date(p_procd_dt)
- Start Date(p_strtd_dt)
- Voided Date(p_voidd_dt)
- Object Version Number(p_object_version_number)

RESULT

The Life Event status is set to 'Processed'.

STEP 9: CREATE POTENTIAL LIFE EVENT FOR CONVERSION LIFE EVENT

The api used is `ben_ptnl_ler_for_per_api.create_ptnl_ler_for_per`

The final step in the conversion process is using this API to create a new Potential Life Event record for the same person with a new Life Event of “Conversion”. This is exactly similar to Step 1, but with a different Life Event. This new Life Event will be processed by OAB’s enrollment process in a later step.

REQUIRED VALUES

- Life Event Occurred Date (`p_lf_evt_ocrd_dt`)
- Potential Life Event Reason for Person Status Code (`p_ptnl_ler_for_per_stat_cd`)
- Enrollment Period (`p_enrt_perd_id`)
- Life Event Reason (`p_ler_id`)
- Person Id (`p_person_id`)
- Business Group Id (`p_business_group_id`)
- Unprocessed Date (`p_unprocd_dt`)
- Effective Date (`p_effective_date`)

OUTVALUES

- Potential Life Event Reason for Person Id(`p_ptnl_ler_for_per_id`)
- Object Version Number(`p_object_version_number`)

RESULT

A new Potential Life Event record with a status of “Unprocessed” is created in the database.

AFTER THE APIS

Once you have mastered and survived the API processing, the true test starts. Unlike HR data conversion where besides verifying, you are finished with the conversion, with OAB you have two more tasks to complete that can cause some heartburn. The following processes tie everything together from the eligibility profile you defined long ago and the program enrollment requirements you configured and weren’t quite sure what some of the codes actually did. This is the true test for both the OAB plan design and the previous HR design and conversion.

RUN THE PARTICIPATION PROCESS

Run the Participation Process for the life event “Conversion” to create the actual Benefit Enrollments. The elections and dependents mapped in using the API's are processed using standard functionality and the plan design set up for your enterprise to determine eligibility, rates, create elements, etc. The API's entered the raw data (legacy elections) that were used within the design of the benefit program to create actual participant enrollments. It is highly recommended to run this in Rollback mode initially with Audit Log set to “Yes” to perform a trial run and identify and correct potential issues before running the process in Commit mode.

If you determine issues once you have run the participation process, you can rollback the process via the concurrent manager. This allows you to change that rate start date code you weren't so sure about when initially configuring the program enrollment requirements.

RUN CLOSE ENROLLMENTS

Run Close Enrollments after running the participation process to set the status of the Conversion life event from Started to Processed. This finishes the process of converting benefits. As suggested before, first run this process in Rollback mode before running the process in Commit mode.

VERIFY AND TEST

As a follow-up for any successful conversion, verify, verify, verify and test, test, test. In addition to running reports to verify legacy to Oracle, also take a large sampling of your participants and verify the enrollment results including elections, rates, element entries, dependents, etc. Also it is very important to perform some of the unit testing scenarios on the converted data. You may discover issues that weren't apparent when testing on sample scenarios.