TRIANZ

When Execution Matters

WHITE PAPER

Oracle Service Agreements Conversion

Author: Ritu Malhotra, Gautam P

October 2006

Table of Contents

1. Executive Summary 3
2. Overview of Contracts 5
2.1. Types of Contracts 5
2.2. Structure 5
3. Key Decisions 7
4. Methodology 10
4.1. Design 10
4.2. Design Tips 10
4.3. Pre Conversion Requirement 11
4.4. Process Flow 12
4.5. Verification of Migrated contracts 13
5. Oracle API's 15
5.1. Contract Header Creation 15
5.2. Line Creation 15
5.3. Sub Line Creation 15
5.4. Billing Schedule Creation and Update 15
5.5. Contract Security Access 16
5.6. Price reflection at Service Line 16
5.7. Security Description and Text 16
5.8. Sales Credit 16
5.9. Standard APIs in SC Module 16
5.10. Core Tables 17
6. Key Challenges 18
6.1. Performance 18
6.2. Tips for Improving Performance 18
6.3. Tax Structure Setup 18
6.4. Billing Schedules Creation and Update 19

Appendix 1 - Contract Structure and Mapping to Tables 20
Appendix 2 – Screen Shots 21
Appendix 3 - Strategy for Rollback and Commit 23
Appendix 4 - Price reflection at Line 24
Appendix 5 - Security Description & Text 25
Appendix 6 – Inventory of Oracle Standard API's 26
Appendix 7 – Inventory of Affected Base Tables 27
Appendix 8 – Inventory of Mandatory Columns 28
Appendix 9 - Inventory of Mandatory Validations 29

1. Executive Summary

In today's business world change is inevitable. Whether due to changes in the business or in business applications, systems are continuously being upgraded. Oracle Enterprise Application solutions are the most widely used applications to manage various business needs. The Oracle Service Contracts ("Oracle SC" or "OSC") module provides a complete contract authoring execution solution to manage warranties, extended warranties, usage, subscription services, as well as complex service agreements. With OSC one can:

- Sell multiple types of services
- Define pricing and billing schedules
- Ensure timely service entitlement checks
- Automate renewals for recurring revenue opportunities
- Simplify change management
- Minimize service revenue leakage.

This document outlines conversions in OSC, which are required when:

- A customer migrates from a lower version of Oracle E-business suite to a higher version, and remodels the business systems.
- Customers migrate data from legacy systems to Oracle Service Contracts when Oracle E-business suite is implemented. Legacy systems may not provide as much functionality/flexibility and this provide many opportunities to restructure the contracts for better manageability and efficiency.

This document provides complete information for importing Oracle Service Contracts and is especially useful because:

- This is one of the first guides available. Standard interfaces/APIS or adequate documentation and support for importing SC data does not currently exist. Implementers have till now relied on private API's (which are not supported by Oracle) or Service Contract Import concurrent program, which has the disadvantages of not only not being a complete solution, but is also lacking support in Oracle versions 11.5.9 or 11.5.10.
- The information here can be used as inputs to solving the challenges in mapping the legacy data with the Oracle SC schema. SC is a powerful application with rich functionality and many choices of setting up the applications and importing the data. Its important that the right decisions are made how to map the convert so that it meets the business needs and also allow enterprise to take advantage of the numerous features to automate the business further
- This document also provides information on dealing with large volumes of data, and verifying imported data to make sure that it's been converted properly.

This will be helpful for BA and implementers who are designing the strategy for conversion and also technical analysts who will actually develop the conversion scripts.

2. Overview of Contracts

2.1. Types of Contracts

A Contract is an agreement to bill a customer for recurring services or extended warranty. This is very common in Service industries.

Briefly, the contracts cycle consists of the following steps. When goods are purchased by a customer, there could be associated services bundled with the goods purchased. The purchased item is tracked in IB and the agreement to provide the service for a fee is tracked in a contract. Contract would track the party with which the agreement is signed and agreed upon terms, conditions, prices, discounts, billing schedule payment terms, renewal options, and fulfillment. Based on the billing schedules customer invoices would be sent out.

Oracle Service Contracts enables users to design, manage, and bill for service offerings tailored to their customers' needs. A contract provides complete solution to manage warranties, subscription and usages. It makes it possible to manage the entire contract life cycle, including contract creation, renewal, and termination. All contracts are held centrally. There are three types of contract:

- Subscription Contract: This is used for tracking tangible and intangible subscription products.
- Warranty and Extended Warranty: It is used when tracking of warranty of device is needed. It needs to be used if tracking of products in customers install base has to be done. It doesn't allow having usage lines and service lines with other level of coverage's.
- Service Agreement: It is used for tracking service entitlements to covered product and Support plans. It needs to be used for contracts which will cover usages and cover items other than customers install base.

2.2. Structure

In Oracle, Service contracts is represented by a Header that contains information about the duration, total value, the parties involved, at what level , price list, renewal terms.

Each contract will have Lines that would represent service, usage or subscription item etc that is covered. Types of Lines are:

- Usage: charges customer based on usage. For e.g.: photo copier machine usage
- Subscription: This covers subscriptions made by customer. For e.g.: yearly subscription of magazine or collateral material send through email.
- Service: this type of line covers broad category of items that can include activities like installation of items, repair.

Each line would have Sub lines that would specify exactly what service covers or the counters when it is usage line. The types of Sub Lines are:

- Usage: This type of sub lines specifies the counter which is used for usage tracking.
- Subscription: It doesn’t allow having any sub lines
- Service: This type of line covers: Party, Customer, Site, System, Product and Item.

3. Key Decisions

There are certain businesses decisions need to be taken before designing the conversion process. A few of them are discussed below:

3.1. Contract number can be automatically generated or can be brought in from legacy system. It would be better if contract number is migrated ASIS from legacy system, as there are usually legal issues involved. For example, invoices sent to customers use particular contract numbers, and if automatic generation of contract number is used then invoices would change after migration.

3.2. Contracts which are not in Active and Signed status need not to be migrated if volume of data is huge. But for legal issues, terminated and expired contracts might need to be migrated.

3.3. This is the stage where data can be cleaned as much as possible. If inactive contacts information is present in ACTIVE contract, we don't need to migrate this instead migrate the correct and active contact information. On similar lines, inactive sales person information can be dropped.

3.4. Older version of contracts and notes information which is no more relevant can be dropped off.

3.5. If active contract is having terminated lines or expired lines even these can be dropped off after considering the legal formalities.

3.6. Any contract attachments which no longer holds good can also be dropped off. If Legacy system is not being dismantled then this can be used for reference.

3.7. Pricing policies - Whether items price would be overridden to the price in the price list. If items price needs to be overridden in contract while migrating them then couple of profile option should be set as given below.

- o OKS: Use Advanced Pricing for Manual Adjustment: should be set to No.
- o OKC: Allow Manual Price Adjustment: should be left as blank.
- o OKC: Enable Advanced Pricing: Should be set to yes

3.8. If billing history of previous invoices sent to customer needs to be brought into new system, then it would be needed to migrate contract starting as same day contract starts in legacy system. If it's not needed then migrate the contract starting as date on which migration is done and store the actual start date in DFF. Again here legal decisions need to be considered.

3.9. Some times contracts are created for 15-20 years even though they would be initially valid for 3-4 years. Instead of migrating contracts for duration of 15-20 years, reduce duration and migrate them and latter use standard Oracle contracts renewal programs.

3.10. Future contract renewal policy needs to be decided. Depending on this decision, renewal rules for contract need to be decided and migrated. For e.g.: By setting the renewal type of a contract to “Active Contract”, the contract is renewed in a status of Active status without any approvals. Independent condition, process definition needs to be created. Concurrent program Date Assembler needs to be scheduled. This concurrent program keep checking how many contracts are due for renew based on the conditions created and it renews the contract based on the process created.

3.11. Need to decide on invoicing rule, accounting rule and payment terms depending on when amount needs to be interfaced to AR and when revenue should be recognized. For e.g.: If the customer needs to be billed at the start of each month, choose Advance Invoice as the Invoicing Rule for the contract.

3.12. The type of billing type to be used needs to be examined.
   - Top Level: Default schedule at the line level - If top level is used, then total amount and amount for all individual months get calculated out of box. But when we use this top level, then its probable that the amount for every month would not the same as monthly unit price. For e.g. If for every month $100 needs to be charged, then Top level might not help and Equal Amount billing should be used.
   - Equal Amount: To bill for equal amounts across sub lines over the billing period and have control over the amount.

3.13. The structure of billing schedules needs to be decided based on how customer would be billed. A billing schedule determines when the customer is billed for the services they receive. For e.g.: Billing schedules with irregular period needs to be handled or not. Billing schedules with irregular period would be required for a customer who starts a contract in the middle of a month, but wished to be billed at the beginning of each month. If customer is ok with billing from middle of the month to middle of the next month, then it would be regular period handling.

3.14. Contracts can be merged, instead of migrating them as multiple contracts with few lines, when:
   - Contracts have same bill to address for a customer.
   - Start dates and end dates of a contract are same for same bill to.

Some of the key benefits of merging are:

- o Performance improvement can be achieved
- o Easy maintainability and renewals.
- o Data in one place/contract.

3.15. The level of coverage to use depends on the type of service customer are selling and on the different needs of enterprise. Type of line and sub lines gets decided based on this. For e.g. If customer covers a broad category of items that can include activities such as field service, depot repair, call center, technical support, or any other user-defined business activities then type of line would be "Service". The following table provides suggested guidelines. It lists some of the main business uses and lists the recommended covered levels for each.

Business Use	Recommended Levels	Examples
Servicing a customer product	• Product	• A washing machine manufacturer sells extended warranties for the products it has sold. • Any serialized product • A fire safety equipment vendor sells service for emergency generators installed in a building.
Servicing high volume and low cost items	• Party • Customer • Site • System	The fire safety equipment vendor services fire sprinklers by customer site because they are too numerous to track in a contract individually.
Providing services not tied to any specific product such as telephone support or safety inspections.	• Party • Customer • Site	Sometimes a service covers a business process rather than any specific product a customer owns. For example: • A computer manufacturer provides telephone support to troubleshoot all of its products. • A vendor of fire protection equipment offers periodic safety inspections at a particular site or building.
Providing support for products the customer plans to purchase in the future.	• Item	Use the Item level to cover products not yet in the installed base. For example, a customer plans to purchase additional generators in the near future so they wish to include them in the same service agreement in order to receive a discount.

4. Methodology

4.1. Design

This section details the methodology for converting the Service agreements, which are associated or not associated to the Installed Products. The steps are as follows:

4.1.1. Extract legacy data/Oracle old version into ASCII flat file.

4.1.2. Create Staging tables to hold legacy data. Additional columns can be created to hold ids for the same columns. Stamping id’s for the columns during the validate process will give improvement in performance.

4.1.3. Create a validation process which will ensure validity and integrity of inbound data. This process can include a set of programs that will validate and additionally stamp the column id’s that are derived for validation. This id’s will be used by Oracle Service Contracts APIs. The errors encountered during the validation phase can be captured in the same staging table itself or separate custom error processing table.

4.1.4. Once the validation process is completed, the same program will pass the validated data/id to the appropriate Service Contracts APIs, which then create service agreements in Service Contracts module. The errors encountered during the process can also be recorded into the staging table itself or Custom Error processing tables.

4.1.5. Validation and Processing errors should be corrected and processed in the subsequent runs.

4.2. Design Tips

4.2.1. Any additional information can be captured in Notes.

4.2.2. Additional information at line, sub line level can be captured by defining context sensitive DFF.

4.2.3. Security level additional information can be stored at security text and description.

4.2.4. Transaction type selected for a contract would appear as source in AR transactions.

4.2.5. Billing schedules can be created by using Billing profile options or by using billing streams. Use first method for setting up simple periodic billing schedules, for example, when a customer orders a year of service and wishes to be billed monthly or quarterly. Using second method would allow creating complex billing schedules or billing schedules with irregular periods, for example, when a customer orders additional service in the middle of a billing period or requires service only intermittently. And moreover, a billing stream is designed to speed up the creation of a set of regular billing periods.

4.2.6. OKS: Billing Schedule Level profile option must be set to Billing type which would be used. It can be either ‘Top Line’ or ‘Equal Amount’.

4.2.7. Sales Credits either at header level or line level is mandatory. If Data extract doesn’t provide then default it to sales person of the Vendor.

4.2.8. Customized QA check Processes can be created if some extra validation needs to be done. Default QA checklist name is ‘DEFAULT QA CHECK LIST’.

4.2.9. Default workflow Process name is ‘K_APPROVAL_PROCESS’ and workflow Name is ‘OKCAUKAP’. If different approval processes is needed then customized version can be created and be used.

4.2.10. If huge volume of data needs to be migrated then performance criteria should be always kept in mind. Fine tuning queries& code is a must as most of the Oracle Contracts API’s are not meant for handling bulk data as it hampers migration performance. Please refer section 1.6 for more information on performance tips.

4.3. Pre Conversion Requirement

The following setup is required in Oracle Applications before running conversions:

4.3.1. All the legacy/Oracle Older version Service Line and sub line items should be defined in inventory.

4.3.2. Legacy/Oracle Older version items and install base records should have been imported in install base. Items are “Service Contract Enabled”.

4.3.3. All the legacy /Oracle Older version ‘Systems’ should be defined in Oracle installed Base.

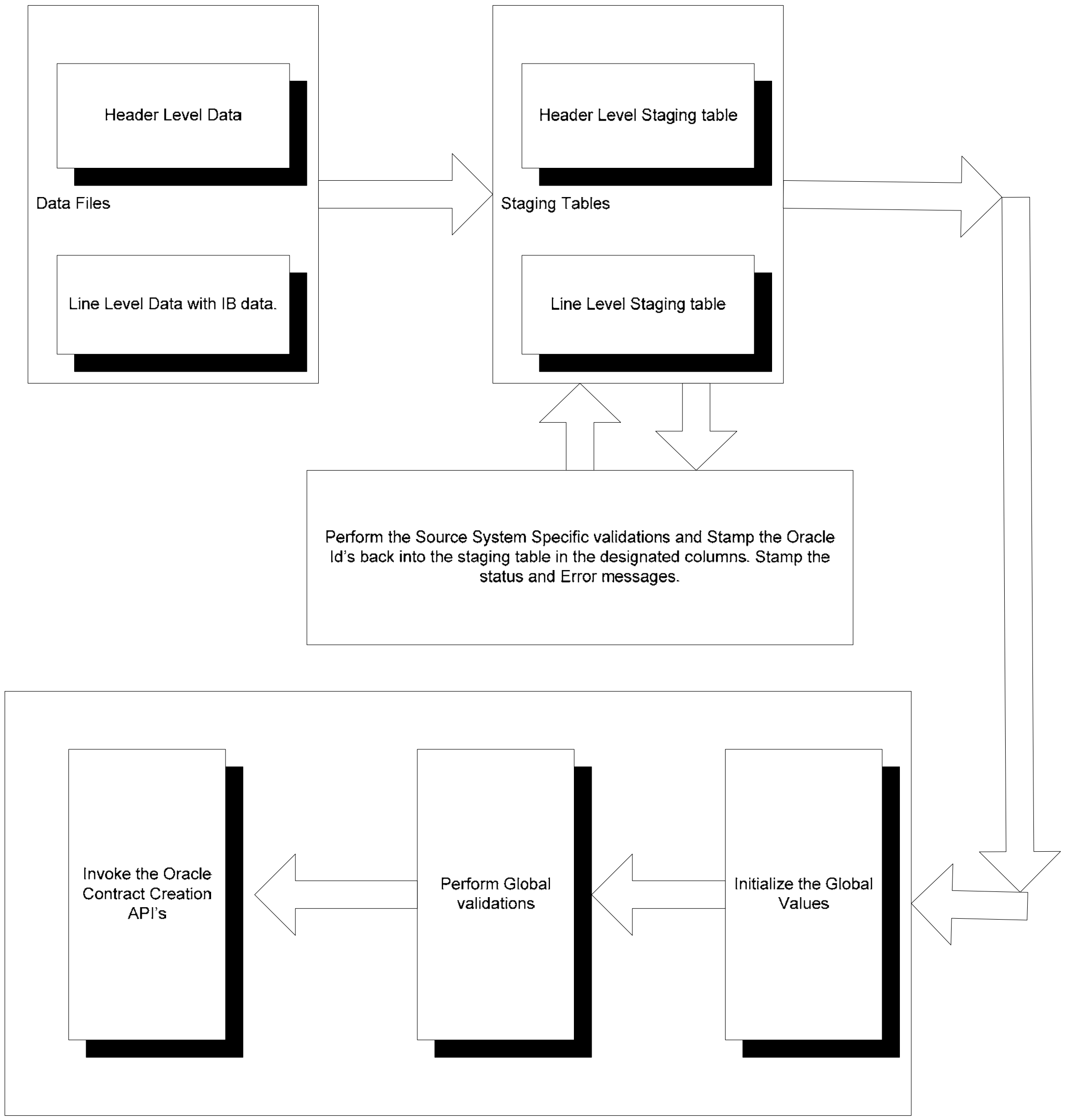
4.3.4. Legacy/Oracle Older version Customer, Accounts, Sites and Contacts should have been already imported in Oracle Receivable

4.3.5. Sales Representatives information needs to be migrated or set up in the system.

4.3.6. The following setups should be completed in the Oracle Service Contracts module
- Time Unit of Measure
- Coverage Templates

4.4. Process Flow

4.4.1. Schematic


Header Level Data
Data Files
Line Level Data with IB data.
Header Level Staging table
Staging Tables
Line Level Staging table
Perform the Source System Specific validations and Stamp the Oracle Id's back into the staging table in the designated columns. Stamp the status and Error messages.
Invoke the Oracle Contract Creation API's
Perform Global validations
Initialize the Global Values

4.4.2. Process Flow Details

After all system validations are done successfully, we would be inserting data into Oracle service contracts base tables in below sequence:

1. Pick contract header record from staging table and create header record. While header record is created, we can also take care of below requirements:
    a) Create sales credit at contract header level
    b) Add contacts to vendor and customer both party levels.
    c) Add contract created to different security groups and resources.

2. If header record is created successfully then pick first line and create it. While line record is created, we can also take care of below requirements:
    a) Line level bill to account and ship to account can be overridden. We can use other values of bill to and ship to account apart from the values which are entered at header level.
    b) Sales credits at service line level can be created.

3. Pick up first sub line which needs to be created under above service line created and create a sub line. While creating sub line, we can associate IB instances also. It depends on the type of sub line. Sub lines specify what service covers or counters where usage is tracked. If line is of type Service then type of sub lines which can be created are:
    a) Party
    b) Customer
    c) Site
    d) System: This type of sub line covers the complete system.
    e) Product: This type of sub line needs an association with IB.
    f) Item

    Process all the sub lines for line created above.

4. Create or update billing schedule for the line created successfully in step 2.

4.5. Verification of Migrated contracts

Once contracts are successfully migrated, they need to be validated and verified. It can be done by:

1. View and verify contracts information manually.

2. Run QA process on the contract.

3. Run Service contracts main billing program and Master auto invoice import program on few of the contracts for a customer. Verify created AR

transactions manually for Revenue recognized, monthly amount for the customer and GL dates.

4. Verify by generating an invoice for a customer after running Print Draft consolidate program. Check billed amount for the customer is correct or not.

5. Manually renew few contracts and verify whether renewal data in a contract is valid.

6. Terminate couple of sub lines or lines and verify the data.

7. Run standard and custom reports on contracts and verify the results with legacy system reports output data.

5. Oracle API's

This section talks about main and important Oracle APIs which would be called for migration.

5.1. Contract Header Creation

For creating contract header API is:
OKS_CONTRACTS_PUB.CREATE_CONTRACT_HEADER

5.2. Line Creation

For creating line of type Service API is:
OKS_CONTRACTS_PUB.CREATE_SERVICE_LINE

5.3. Sub Line Creation

If Item is IB track able then we can create covered product sub line with IB link established. API is:
OKS_CONTRACTS_PUB.CREATE_COVERED_LINE

If item is non IB track able then we can create sub line of Item type. For creating covered customer sub line, there is no single API which can be invoked. We need to invoke group of APIs as listed below:

OKC_CONTRACT_PUB.CREATE_CONTRACT_LINE
OKS_TAX_UTIL_PVT.GET_TAX
OKS_CONTRACT_LINE_PUB.CREATE_LINE
OKC_CONTRACT_ITEM_PUB.CREATE_CONTRACT_ITEM

5.4. Billing Schedule Creation and Update

This is the area where there is least documentation available and it is the most important part of a Service contract. Once all sub lines are created for a particular line, we need to create billing schedules. API is: OKS_CONTRACTS_PUB.CREATE_BILL_SCHEDULE. This API creates billing schedules for both sub line and service line. It creates billing streams and individual billing lines.

For updating billing schedules, there is no API. We can use below API: OKS_BILL_SCH.CREATE_BILL_SCH_RULES.

This API also creates billing schedules for both sub line and service line. It creates billing streams and individual billing lines both. But this API deletes all the billing streams and individual billing first if already present and then creates new ones. Please note that there is no separate public API to create billing schedules just for sub line. This API creates/updates billing schedules for both lines: at sub line and service line.

5.5. Contract Security Access

To add contract created to different security groups and resources API is: OKC_CONTRACT_PUB.CREATE_CONTRACT_ACCESS.

5.6. Price reflection at Service Line

The API which creates sub lines doesn't update amount at line automatically. It needs to be done explicitly by calling API: Oks_Qp_Int_Pvt.COMPUTE_PRICE. This API computes the price and updates the price at service line. Depending on the input parameter Intent, this API decides how price is computed at service line. For Intent values refer Appendix.

5.7. Security Description and Text

To add security description and text under Contracts header Security /Text tab as shown in below screen shot, there is no API. We will have to call direct update statement to update these fields when contract header is created. Refer appendix for update statement.

5.8. Sales Credit

For creating sales credit separately, we will have to use API: OKS_SALES_CREDIT_PUB.INSERT_SALES_CREDIT.
This API can be used to create Sales credits at header level for already created Contract. And even sales credits at line level can be created when line is already created. If only contract header id is passed and line id input is not passed as input parameter to the API then sales credits is created at Contract header level. If both contract header id and line id input is passed as input parameters to the API then sales credits is created at Contract line level.

5.9. Standard APIs in SC Module

- OKC_CONTRACT_PUB (headers, lines, sub-lines) – This is used for creating Headers etc. Please refer section 1.8.6 for further details.
- OKC_RULE_PUB (rule groups, rules)
- OKC_CONTRACT_ITEM_PUB (items)
- OKC_CONTRACT_PARTY_PUB (customers, contacts)
- OKC_CONTRACT_GROUP_PUB (groupings)
- OKS_SALES_CREDIT_PUB (sales credits)
- OKS_CONTRACTS_PUB (billing schedule)

5.10. Core Tables

Please refer section 1.8.7 for association with above APIs.

- ❑ OKC.OKC_K_HEADERS_B
- ❑ OKC_K_HEADERS_TL
- ❑ OKC.OKC_K_LINES_B
- ❑ OKC.OKC_K_LINES_TL
- ❑ OKC.OKC_RULE_GROUPS_B
- ❑ OKC.OKC_K_RULES_B
- ❑ OKC.OKC_K_ITEMS
- ❑ OKC.OKC_K_GRPINGS
- ❑ OKC.OKC_K_PARTY_ROLES_B
- ❑ OKC.OKC_K_HISTORY_B
- ❑ OKC.OKC_K_PROCESSES
- ❑ OKS.OKS_K_SALES_CREDITS
- ❑ OKS.OKS_LEVEL_ELEMENTS

6. Key Challenges

6.1. Performance

This is the biggest challenge for contracts conversion. There is no bulk API's to be used. So, one need to be very careful while deciding the methodology of conversion.

6.2. Tips for Improving Performance

6.2.1. Billing schedule creation

While converting a contract, if performance is bad, then most of the times it is due to the time consumed while creating Individual billing schedules. Conversion performance improves if we create the billing schedule once for one service line i.e. first create service line and all its sub lines, and in the end create billing schedules for the service line and all its sub lines in one go by calling API: OKS_CONTRACTS_PUB.CREATE_BILL_SCHEDULE.

If needed billing stream amounts can be stamped in staging table as and when sub line is created. This approach improves performance drastically if contract is for long duration and with lot of IB associations. We can even create billing schedule once when first sub line is created and then create all sub lines and in the end call update billing schedule API. This approach avoids calculating of all id's and parameters which are needed for creating billing schedules. It's like you have parameters, so create billing schedules once. And in the end when all sub lines are created for a particular service line then update billing schedules. Update API doesn't need as many input parameters as Create billing schedule needs.

6.2.2. Strategy of commit and rollback selected also controls performance.

6.2.3. Deciding on what data can be dropped off can even give performance improvement such as dropping of attachments.

6.2.4. Stamping ID's during validation phase.

6.2.5. Billing history is being brought forward in new system or not. Reducing the contract duration by not bringing in schedules for past duration would help in drastic performance improvement.

6.3. Tax Structure Setup

Tax set up needs to be done before contracts conversion is started. Lots of API’s will through errors due to incorrect set up of tax structure. And some times error message would be quite different when compared to the actual error.

6.4. Billing Schedules Creation and Update

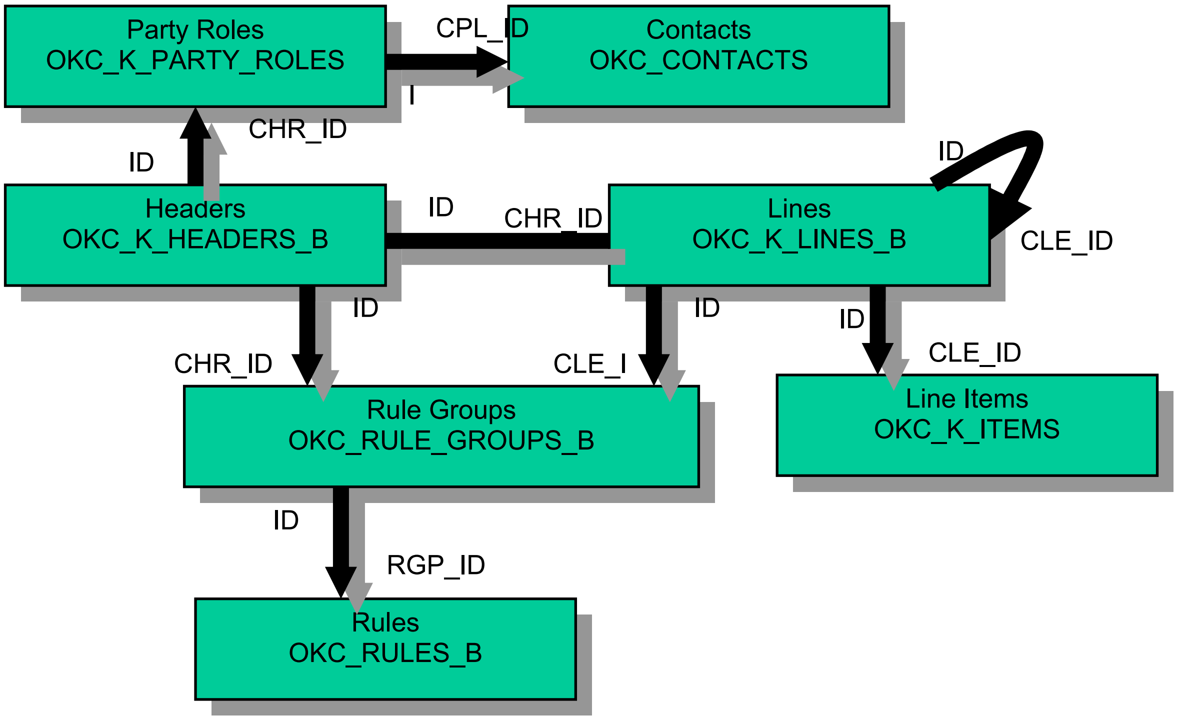
This is the area where there is least documentation available and it is the most important part of a Service contract. Once all sub lines are created for a particular line, we need to create billing schedules through the API:
OKS_CONTRACTS_PUB.CREATE_BILL_SCHEDULE.
This API creates billing schedules for both sub line and service line. It creates billing streams and individual billing lines.

For updating billing schedules, we can use:
OKS_BILL_SCH.CREATE_BILL_SCH_RULES.
This API also creates billing schedules for both sub line and service line

The only difference between the two API’s is that the second API always deletes already existing billing schedules if present and then creates fresh ones. When schedule is already created and if you call OKS_BILL_SCH.CREATE_BILL_SCH_RULES for updating then what would happen is it would create additional billing streams and one would get to see two billing streams instead of a single one.

Appendix 1 - Contract Structure and Mapping to Tables

The diagram below shows the contract table names and how they are linked to each other.



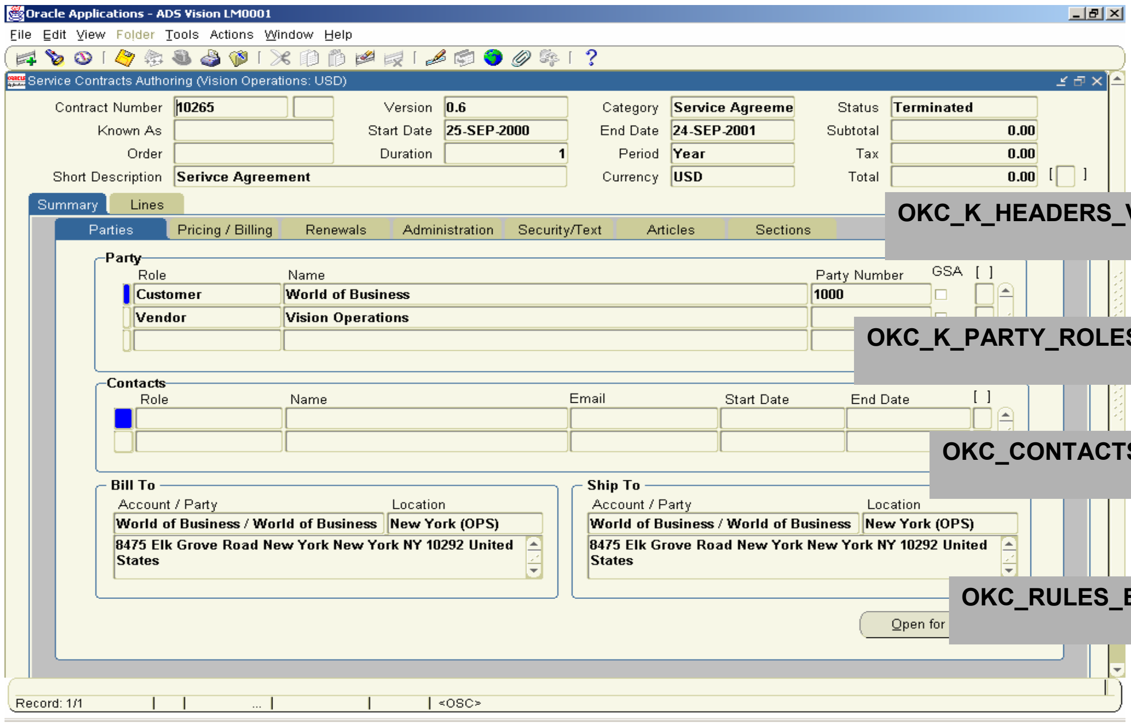
Some important column names are:

- ID: Primary Key in a table
- CLE_ID: Parent Line ID
- DNZ_CHR_ID: De-normalized Contract ID
- JTOT_OBJECT1_CODE: Object Code from JTF_OBJECTS_VL table. Contains the SELECT statement which retrieves the data
- OBJECT1_ID1, OBJECT1_ID2: Keys to access the object in JTF_OBJECTS_VL

Appendix 2 – Screen Shots

Below screen shots will give an idea which all tables are linked for which data:

1. Contract Header Screen shot and its related tables:



Contract Number	10265	Version	0.6	Category	Service Agreeme	Status	Terminated
Known As		Start Date	25-SEP-2000	End Date	24-SEP-2001	Subtotal	0.00
Order		Duration	1	Period	Year	Tax	0.00
Short Description	Serivce Agreement			Currency	USD	Total	0.00

Party

Role	Name	Party Number	GSA
Customer	World of Business	1000	
Vendor	Vision Operations		

Contacts

Role	Name	Email	Start Date	End Date

Bill To

Account / Party	Location
World of Business / World of Business	New York (OPS)

8475 Elk Grove Road New York New York NY 10292 United States

Ship To

Account / Party	Location
World of Business / World of Business	New York (OPS)

8475 Elk Grove Road New York New York NY 10292 United States

- OKC_K_PARTY_ROLES_B
    - If JTOT_OBJECT1_CODE = 'OKX_PARTY' join to HZ_PARTIES
    - If JTOT_OBJECT1_CODE = 'OKX_OPERUNIT' join to HR_ORGANIZATIONS

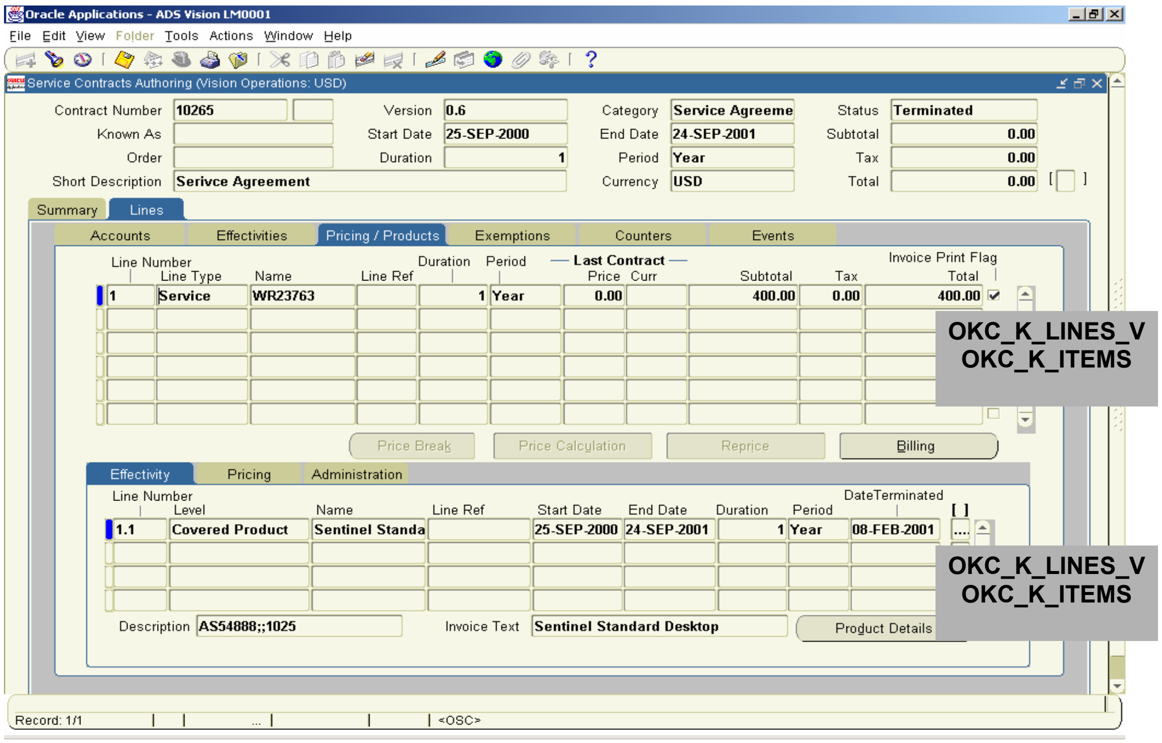
- Header Rules stored in OKC_RULES_B

- OKC_RULES_B -> RULE_INFORMATION_CATEGORY defines what kind of rule it is (eg. STO = Ship To, BTO = Bill To, REN = Renewal)
- All rules defined in "OKC Rule Developer DF" Descriptive Flex field definition

- OKC_RULES_B
    - JTOT_OBJECT1_CODE, OBJECT1_ID1, OBJECT1_ID2 store information on the rules that are stored in other tables (eg. Price List)

- RULE_INFORMATION1 to RULE_INFORMATION15 store specific details about a rule that is not available in other tables (eg. Estimated Percent)

2. Contract Lines and Sub Lines Screen shot and its related tables:



Contract Number 10265
Version 0.6
Category Service Agreeme
Status Terminated
Known As
Start Date 25-SEP-2000
End Date 24-SEP-2001
Subtotal 0.00
Order
Duration 1
Period Year
Tax 0.00
Short Description Serivce Agreement
Currency USD
Total 0.00

Line Number	Line Type	Name	Line Ref	Duration	Period	Last Contract Price	Curr	Subtotal	Tax	Total	Invoice Print Flag
1	Service	WR23763		1	Year	0.00		400.00	0.00	400.00	✓

Line Number	Level	Name	Line Ref	Start Date	End Date	Duration	Period	DateTerminated
1.1	Covered Product	Sentinel Standa		25-SEP-2000	24-SEP-2001	1	Year	08-FEB-2001

Description AS54888;;1025
Invoice Text Sentinel Standard Desktop

Appendix 3 - Strategy for Rollback and Commit

We can have different approaches for commit and rollback. Strategy selected controls performance also to some extend. Best recommended way would be to commit only if the complete contract is created successfully. Header, all its lines and all its sub lines are created successfully then commit for a particular contract. If any one of them fails it would be best to rollback. This helps in achieving maximum integrity of data created or migrated.

Appendix 4 - Price reflection at Line

The API which creates sub lines doesn't update amount at line automatically. It needs to be done explicitly by calling API: Oks_Qp_Int_Pvt.COMPUTE_PRICE. Depending on the input parameter Intent, this API decides how price is computed at service line.
Intent can take below values:

a. HP: Header Pricing. When intent is HP, it calculates the price for complete contract at header level.
b. LP: Top Line Pricing. This computes price at line level. Service line id would need to be passed as input parameter.
c. SP: Sub Line Pricing. This computes price at sub line level. Sub line id would need to be passed as input parameter.
d. OA: Override Pricing. This would be used, when we are overriding price of price provided in price list. It's like sub line Sub total is overridden and price needs to be computed at its line level, then this intent would need to be passed.
e. SB_P: Subscription Pricing
f. SB_O: Subscription Override Pricing.

Appendix 5 - Security Description & Text

There is no API to populate Security Description and text. Direct update statement would be required, such as:

```
UPDATE okc_k_headers_tl
SET description = Security Description from data extract,
Comments = Security Text from data extract
Where ID = ID of contract created.
```

Appendix 6 – Inventory of Oracle Standard API’s

No.	Oracle Standard API	Description	Comments
1	OKS_CONTRACTS_PUB.CREATE_CONTRACT_HEADER		To Create Contract Header
2	OKC_CONTRACT_PUB.CREATE_CONTRACT_ACCESS		To Provide Read or Modify Access to Contract
3	OKS_CONTRACTS_PUB.CREATE_SERVICE_LINE		To Create Service Line
4	OKS_CONTRACTS_PUB.CREATE_COVERED_LINE		To Create Covered Product Sub line
5	OKC_CONTRACT_PUB.CREATE_CONTRACT_LINE, OKS_TAX_UTIL_PVT.GET_TAX, OKS_CONTRACT_LINE_PUB.CREATE_LINE, OKC_CONTRACT_ITEM_PUB.CREATE_CONTRACT_ITEM		To Create Covered Customer
6	OKS_CONTRACTS_PUB.CREATE_BILL_SCHEDULE		To Create Billing Schedule

Appendix 7 – Inventory of Affected Base Tables

Schema	Base Table	Description	Comments
OKC	OKC_K_HEADERS_B	OKS_CONTRACTS_PUB.CREATE_CONTRACT_HEADER	To Create Contract Header
OKC	OKC_K_ACCESSES	OKC_CONTRACT_PUB.CREATE_CONTRACT_ACCESS	Insert Access Privileges to the Table
OKC	OKC_K_LINES_B	OKS_CONTRACTS_PUB.CREATE_SERVICE_LINE	Creates Service Line
OKC	OKC_K_LINES_B	OKS_CONTRACTS_PUB.CREATE_COVERED_LINE	Creates Covered Lines Distinguished by lse_id
OKC	OKC_K_LINES_TL	OKC_CONTRACT_PUB.CREATE_CONTRACT_LINE	
OKC	OKS_STREAM_LEVELS_B , OKS_LEVEL_ELEMENTS	OKS_CONTRACTS_PUB.CREATE_BILL_SCHEDULE	Create Billing Schedule

Appendix 8 – Inventory of Mandatory Columns

Interface table / API name	Column Name	Comments
OKS_CONTRACTS_PUB.CREATE_CONTRACT_HEADER	customer_id , Bill and Ship To Location ID , Pricelist ID ,Contract Number and Contract Name	Depending on the profile set up , Contract number may be passed or obtained automatically
OKC_CONTRACT_PUB.CREATE_CONTRACT_ACCESS	Contract ID	To Provide Read or Modify Access to Contract
OKS_CONTRACTS_PUB.CREATE_SERVICE_LINE	Contract Id , Line Type , Currency ,UOM Code	To Create Service Line
OKS_CONTRACTS_PUB.CREATE_COVERED_LINE	Contract_id , service_line_id , ib_instance_id ,quantity , UOM Code , Currency Code	Creates Covered Line LSE Id = 9
OKC_CONTRACT_PUB.CREATE_CONTRACT_LINE	Negotiated Amount , Line Number , Service Line ID	To Create Covered Customer with LSE id 35
OKS_CONTRACTS_PUB.CREATE_BILL_SCHEDULE	dnz_chr_id , cle_id, level_periods, uom_per_period	

Appendix 9 - Inventory of Mandatory Validations

No.	Validation Column Name	Comments
1	Check the UOMs	Check OKX_UNITS_OF_MEASURE_V for UOM (Day ,Month , Each)
2	Verify Contract Header Status	OKC_STATUSES_TL WHERE UPPER(MEANING) = UPPER(l_contract_header_status)
3	Verify the Contract Group	FROM OKC_K_GROUPS_V WHERE UPPER(NAME) = UPPER(l_contract_group);
4	Verify the Currency Code	SELECT 1 FROM FND_CURRENCIES WHERE UPPER(CURRENCY_CODE) = UPPER(l_currency_code);
5	Verify the Workflow Process	FROM OKC_PROCESS_DEFS_B WHERE UPPER(WF_PROCESS_NAME) = UPPER(l_workflow_process);
6	Verify the Price List	FROM QP_LIST_HEADERS_tl a, QP_LIST_HEADERS_b b WHERE UPPER(a.NAME) = UPPER(trim(l_pricelist_name))
7	Verify the Accounting Rules	FROM RA_RULES WHERE UPPER(NAME) = UPPER(l_accounting_rule)
8	Verify the Invoicing Rules	FROM RA_RULES WHERE UPPER(NAME) = UPPER(l_invoicing_rule)
9	Verify the Payment Terms	FROM RA_TERMS WHERE NAME = l_payment_terms;
10	Verify the Customer	FROM HZ_PARTIES WHERE UPPER(PARTY_NAME) = UPPER(l_customer_name)
11	Verify the Renewal Type	FROM FND_LOOKUPS WHERE LOOKUP_TYPE = 'OKC_RENEWAL_TYPE' AND MEANING= _renewal_type;
12	Verify Line Renewal Type	FROM FND_LOOKUPS WHERE LOOKUP_TYPE = 'OKC_LINE_RENEWAL_TYPE' AND MEANING= l_renewal_type;
13	Verify Contract Line Status	FROM OKC_STATUSES_TL WHEREUPPER(MEANING) = UPPER(l_contract_line_status)

No.	Validation Column Name	Comments

14	Check Sales Person	FROM okx_salesreps_v
		WHERE NAME IN(l_sales_rep)

15	Check Resources	ROM OKX_RESOURCES_V WHERE NAME = l_contact_name

16	Check Party Contact	FROM OKX_PARTY_CONTACTS_V
		WHERE NAME = l_contact_name

17	Check Items	FROM OKX_SYSTEM_ITEMS_V
		WHERE DESCRIPTION IN

18	Check Contract Already Exist	FROM okc_k_headers_b
		WHERE contract_number =
		l_contract_number;

19	Check Resource	FROM OKC_RESOURCE_USERS_V
		WHERE NAME = l_res_name

20	Check Resource Group	WHERE NAME = l_group_name AND RESOURCE_TYPE = 'RS_GROUP'

21	Check Contact Roles	FROM FND_LOOKUP_VALUES
		WHERE LOOKUP_TYPE =
		'OKC_CONTACT_ROLE'
		AND UPPER(MEANING) =
		UPPER(l_role_name)

22	Contract Start Date and End Date (Both at Header and Line Level) Cant be Null and End Date Should be Greater than Start Date and Date should be in Proper Format	

About the Author

Ritu Malhotra, a Lead Consultant at Trianz, has more then six years of experience in implementing various Oracle applications. Her expertise is in Oracle Service Contracts, Teleservices, OM and Receivables. She has successfully led implementation of several Oracle products at companies such as Oracle, VeriSign, Good Technologies and Equinix. She can be contacted at ritu.malhotra@trianz.com

About Trianz

Trianz is a global consulting and professional services firm that helps leaders successfully execute on business and technology initiatives to achieve results as measured from a top management perspective. We provide a wide range of management consulting, technology and engineering and operations outsourcing services for a diversified global client base.

Trianz clients are results-focused executives and leaders in a range of organizations from Fortune 1000 corporations to emerging, rapid-growth companies. Trianz seeks to be the execution partner of choice for our clients and a leading participant in their global strategic initiatives. Our service offerings focus on the following areas:

- Operations Consulting
- Enterprise Applications Services
- Software Product Engineering
- Transformational Outsourcing