

BPEL SDL Design

Creation Date: December 9, 2013

Last Updated: June 20, 2014



The information in this document is proprietary to Neustar, Inc. It may not be used, reproduced, nor disclosed without explicit written approval.

Table Of Contents

[Document History 4](#_Toc391066223)

[Document Approvals 4](#_Toc391066224)

[1 Document Overview 5](#_Toc391066225)

[1.1 Introduction 5](#_Toc391066226)

[1.2 ASSUMPTIONS 6](#_Toc391066227)

[1.3 Related Documents 6](#_Toc391066228)

[1.4 Issue Log 7](#_Toc391066229)

[2 Overview 7](#_Toc391066230)

[3 PROCESSING FLOWS 8](#_Toc391066231)

[3.1 PREVALIDATION AND CSR REQUEST SUBMISSION 9](#_Toc391066232)

[3.2 CSR RESPONSE PROCESSING 10](#_Toc391066233)

[3.3 CHILD ORDER CREATION 11](#_Toc391066234)

[3.4 LSR REQUEST SUBMISSION 12](#_Toc391066235)

[3.5 LSR RESPONSE AGGREGRATION 13](#_Toc391066236)

[3.6 SOA REQUEST SUBMISSION 14](#_Toc391066237)

[3.7 SOA RESPONSE AGGREGRATION 15](#_Toc391066238)

[3.8 FULFILLMENT PROCESSING 16](#_Toc391066239)

[4 PREVALIDATOIN AND REQUEST PROCESSING 17](#_Toc391066240)

[4.1 COMPONENT CHANGES 17](#_Toc391066241)

[5 Installation Impact 48](#_Toc391066242)

[6 Product Impact 48](#_Toc391066243)

[6.1 ESR GUI 48](#_Toc391066244)

[6.2 REPORTIN and BILLING 48](#_Toc391066245)

[7 Data Model Changes 48](#_Toc391066246)

[7.1 SEQUENCE 48](#_Toc391066247)

[7.2 SCHEMA CHANGES –SEA SCHEMA 48](#_Toc391066248)

[7.3 Oracle JOB/Schedule Configuration 51](#_Toc391066249)

[7.4 STORED PROCEDURE (SEA DB) 51](#_Toc391066250)

[8 Component Changes 52](#_Toc391066251)

[8.1 Schema CHANGES 52](#_Toc391066252)

[8.2 XSL transformation Templates 52](#_Toc391066253)

[8.3 Process Director 52](#_Toc391066254)

[8.4 JCODE CHANGES 52](#_Toc391066255)

[9 Re-do functionality form ESR GUI 52](#_Toc391066256)

[10 TIER 2 REFLOW SERVICES 52](#_Toc391066257)

[10.2 ORDER stuck in PREVAL SUCCESSFUL EVENT 53](#_Toc391066258)

[10.3 ORDER stuck in CSR\_REJECTED/CSR\_FAILURE event 53](#_Toc391066259)

[10.4 order return mix onsp order 54](#_Toc391066260)

[10.5 CHANGES to existing BPEL composite 55](#_Toc391066261)

[10.6 ASSUMPTIONS 55](#_Toc391066262)

[11 SCENARIOS TO BE CONSIDERED 55](#_Toc391066263)

[11.1 Glossary of Terms 55](#_Toc391066264)

Document ContrOL

## Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Revision Description** | **Author** |
| 10/12/2013 | 1.0 | Initial version | Rajesh Vishwakarma |
| 2/1/2014 | 1.1 | Updated for java components, MLPrevalidationUtil and BpelSynchronization API | Vijay Agalcha/Ankur Shrimali |
| 21/1/2014 | 1.2 | Updated design for name changes of bpel components | Ankur Shrimali |
| 24/2/2014 | 1.3 | Updated with design for Mix ATN support | Ankur Shrimali |
| 5/14/2014 | 1.4 | Updated with SOA IVR, DL Migration and DL scheduling | Ankur Shrimali |
| 6/4/2014 | 1.5 | Updated with DL Migration design | Ankur Shrimali |

## Document Approvals

|  |  |  |
| --- | --- | --- |
| **Department** | **Representative** | **Approval Date** |
| Architecture | George Zangos |  |
| Billing Systems | Teresa Bystrek |  |
| Engineering NeuStar | Madhav Chimurkar |  |
| NeuStar QA | Jeffrey Thiel |  |
| Product Management | Anh Ngo |  |

# Document Overview

## Introduction

The purpose of this document is to describe the enhancements required to the existing Neustar BPEL automation to support Suddenlink requirements.

A custom Adapter will be built to connect to Suddenlink’s current interface. BPEL automation services will be enhanced to support SDL requirements around supporting mix order subtype and multiple ONSP/ATN orders.

BPEL 9.2 Release:

* MIX ONSP support: Support for Suddenlink PORTIN orders having multiple TNs belonging to different service providers. Order will split into multiple child orders for individual CSR and LSR transaction for different ILECs.
* MIX order subtype support: Support for Suddenlink PORTIN orders with NATIVE and PORTED TNs.

BPEL 9.3 Release:

* Multi ATN Support: Support for Suddenlink PORTIN orders where TNs belonging to one service provider will have different ATNs.
* Support for Large order handling: Automation will avoid large payload propagation through BPEL automation. Orders received from Suddenlink will be logged into ORDER\_INFO table and transaction specific information will be retried before performing specific transaction.

BPEL 9.3 Release:

* ESR shall support the SOA activation at the ATN level in addition to the TN level. If a given TN of an order is provided in the IVR, ESR will activate all TNs associated with that given TN.

BPEL 9.3.5 Release:

* Log ComplexMultilineIndicator field in SEA\_ESR\_TRANS table.  
  Automation will populate ComplexMultilineIndicator with 'M' for master order and 'S' for child orders. This field will be 'S' for single line PORTIN orders.

BPEL 9.4 Release:

* DL Migration: support for residential to commercial and commercial to residential migration on the basis of TOS/TOA in the ESR order and the values returned in the CSR/DLI responses

BPEL 9.5 Release:

* DL Scheduling: DL will be submitted as per ILEC schedule and DDD will be adjusted as per configured no of days.
* TIER II services for SDL multiline orders.

## ASSUMPTIONS

|  |  |
| --- | --- |
|  | **Assumptions/Pre-Conditions** |
| 1 | ESR GUI will be updated to handle fallout scenarios for CSR/SOA transactions. |
| 2 | IVR activation is not part of this design document |
| 3 | Automation will submit SvCreateRequest based on the focdate of individual child order. |
| 4 | Confirmation response will be sent with farthest due date from child orders. |
| 5 | BPEL maps will be updated to bypass TXNUM generation for SDL customer. This will be taken care by automation for SDL. |
| 6 | As per new requirement automation will submit E911 request to Intrado for SDL customer. |
| 7 | In case of mix onsp order, first reject on child order will be persisted on master order and same will be sent back to adapter. |
| 8 | Automation will not persist %submitted event on master order. |
| 9 | CSR Import feature: - For multi ATN scenario, ATN will always be overridden in LSR data even if CSR import indicator is N for the combination. |

## Related Documents

|  |  |
| --- | --- |
| **#** | **Document Version and Location** |
| 1 | //custom/Suddenlink/SuddenlinkFlows v5.vsd **(CL#336667)** |
| 2 | //custom/Suddenlink/Suddenlink Requirement.doc **(CL#336687)** |
| 2 | //custom/Suddenlink/Multiple ATN LEC Use Cases.docx **(CL#336596)** |

## Issue Log

Issue log is synced in confluence <https://confluence.nexgen.neustar.biz/display/OMS/Suddenlink+IssueLog>

# Overview

Following are the main objectives of this release:

* Multiple ATNs/LECs support (BPEL 9.2 and BPEL 9.3 release)
  + Currently automation supports PORTIN orders that involve number porting from a single service Provider. SDL will send PORTIN orders where some TN may belong to different Network providers. Also the NPX-NXX information could be different for the TNs within the same provider. This requires that the SDL order will need to be segmented as follows
    - On basis of ONSP.
      * LSR: On basis of ATN
        + SOA: on basis of LRN
      * CSR: On basis of CC / CCNA information
  + Suddenlink may not send correct ATN in the initial request. Automation needs to determine the ATN based on the CSR lookup.
* Mixed order subtype support (BPEL 9.2 and BPEL 9.3 release)
  + Suddenlink can send PORTIN orders with Ported and Native TNs on a single order. Automation needs to perform Porting processing for Ported TNs and NATIVE processing for native TNs. The requirement is to hold FF processing of NATIVE TNs until PORTIN processing is complete.
  + Automation needs to determine if PORTIN TNs are pending for disconnect submission in SEA\_DISCO table. All such TNs will be grouped together as order subtype RECLAIM.
* DL Migration changes (BPEL 9.4 release)
  + DL Migration support for residential to commercial and commercial to residential migration on the basis of TOS/TOA in the ESR order and the values returned in the CSR/DLI responses.

# PROCESSING FLOWS

This section describes how Suddenlink mix-ONSP/mix-ATN requests will be processed. This section also briefs about the new composites that will be created and changes to existing composites. Composite level information is also provided in component changes section.

* Automation will receive Suddenlink request from SDL adapter in SDL\_ESR\_ORDER\_IN\_QUEUE in ESR Order format.
* DequeueSDLRequest (Existing) BPEL composite will de-queue the ESR Order request from SDL\_ESR\_ORDER\_IN\_QUEUE.
* DequeueSDLRequest will invoke svcProcessMLRequest BPEL composite with SEA\_Request.
* svcProcessMLRequest
  + For PORTIN order this composite will invoke svcMLPortinRequest

For non PORTIN orders, this will invoke svcProcessRequest (existing) after receiving the request from SDL adapter automation will log the received order in sea database using data layer.

Automation will perform pre-validations on individual PORTED and NATIVE Tns. The output of pre-validation will be logged into ONSP\_NPA\_NXX\_INFO. Pre-validation process will segment the original SDL order into multiple child orders on the basis of ONSP/NPA/NXX/LRN and CCNA information.

Automation will not persist child orders until CSR processing is complete. Any failure during pre-validation/csr\_failure will be persisted on the original order.

After Pre-validation and CSR processing, automation will start persisting child orders in ORDER/TRANS and HISTORY tables. From here onwards child orders will be processed individually.

Responses on individual child orders will be aggregated by Aggregation composite. After all child orders have received positive responses automation will trigger next transaction on individual child order.

## PREVALIDATION AND CSR REQUEST SUBMISSION

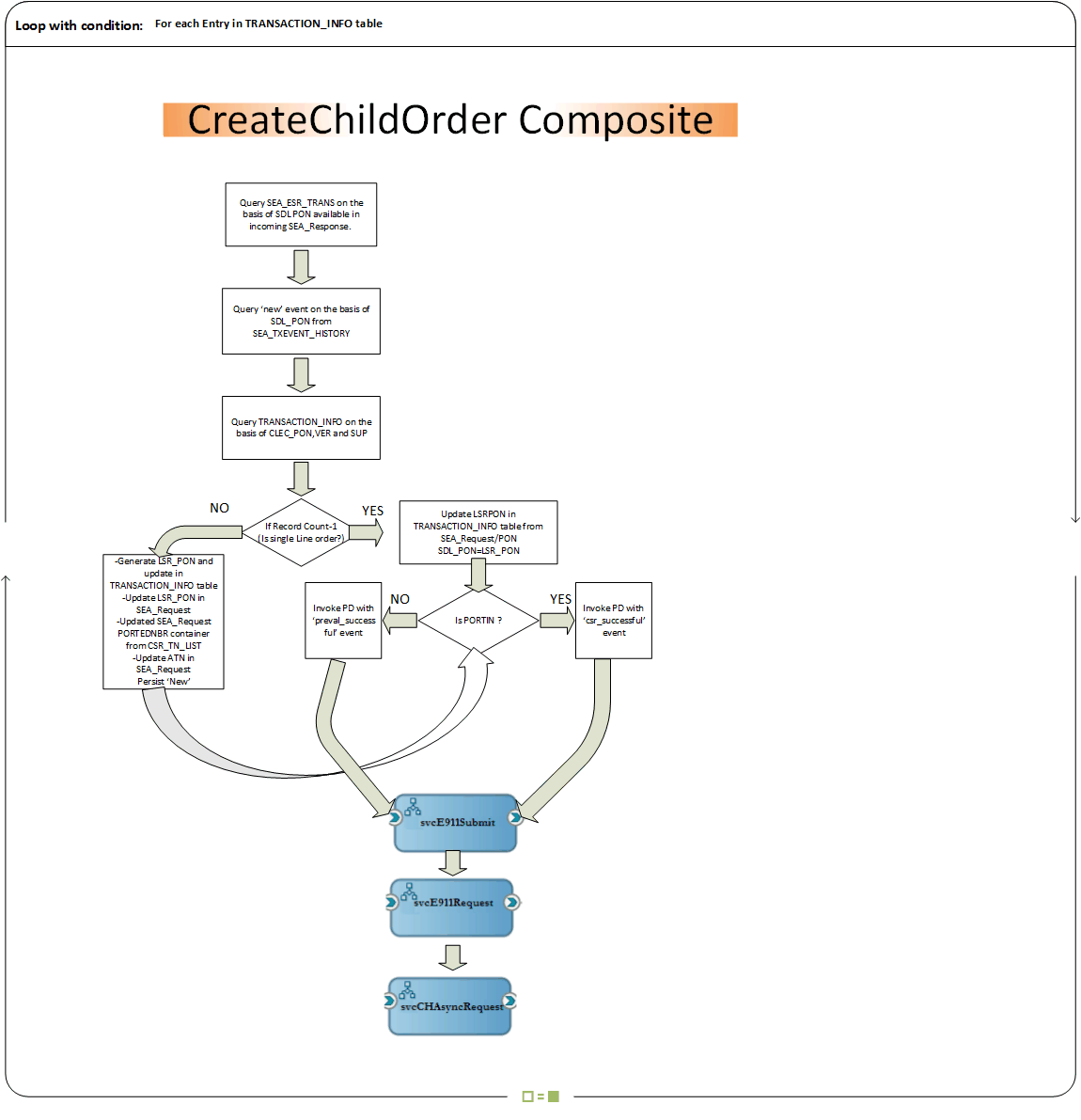


## CSR RESPONSE PROCESSING

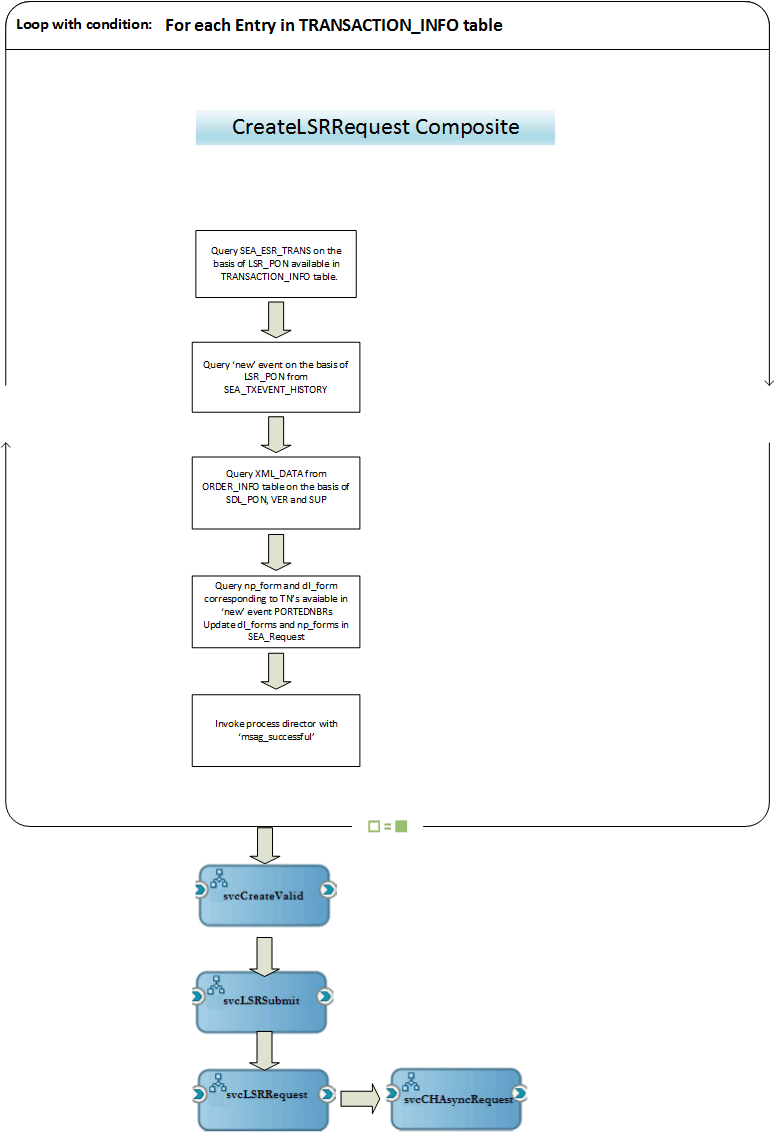


## CHILD ORDER CREATION

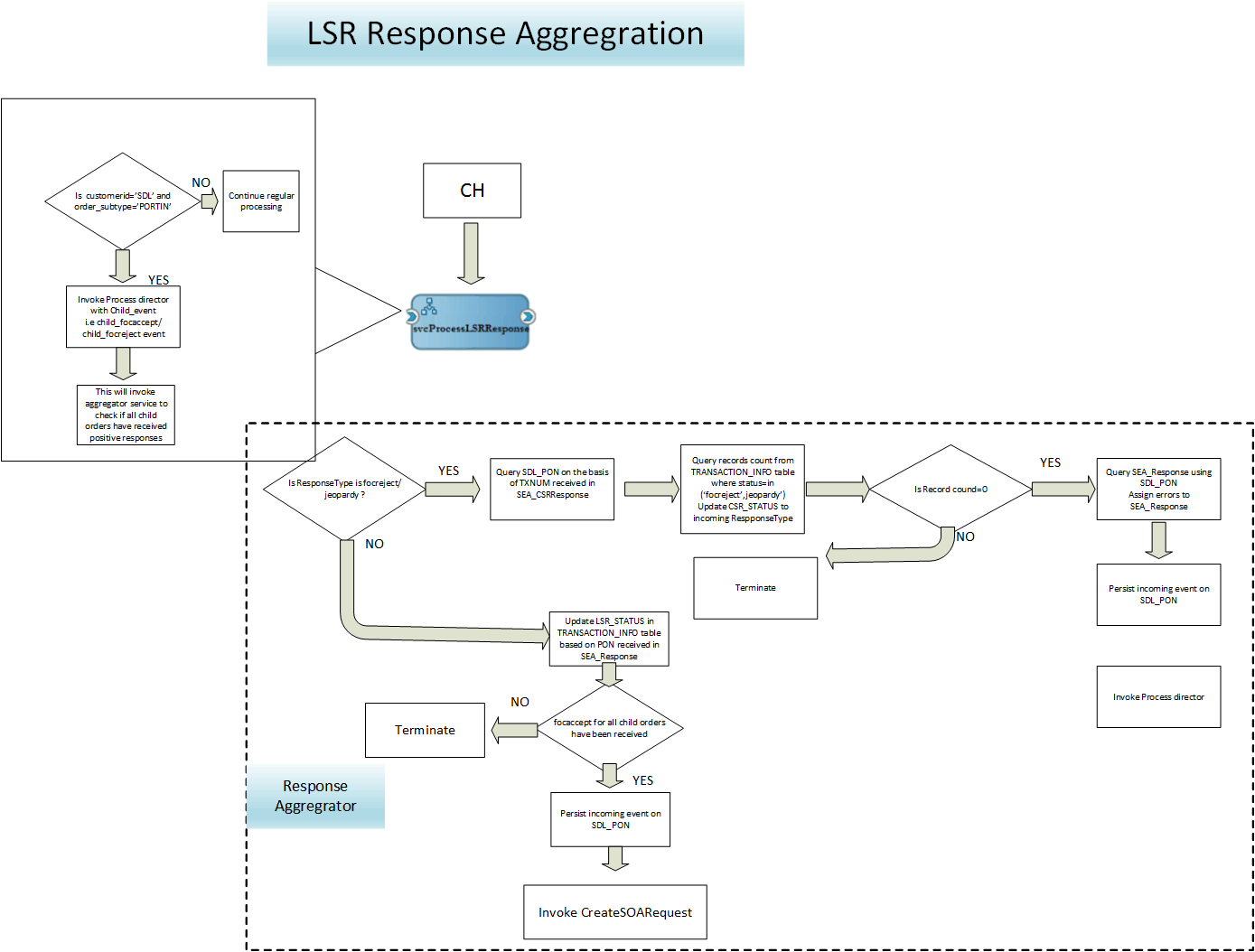
Automation will start creating child orders after receiving CSR responses for individual orders.



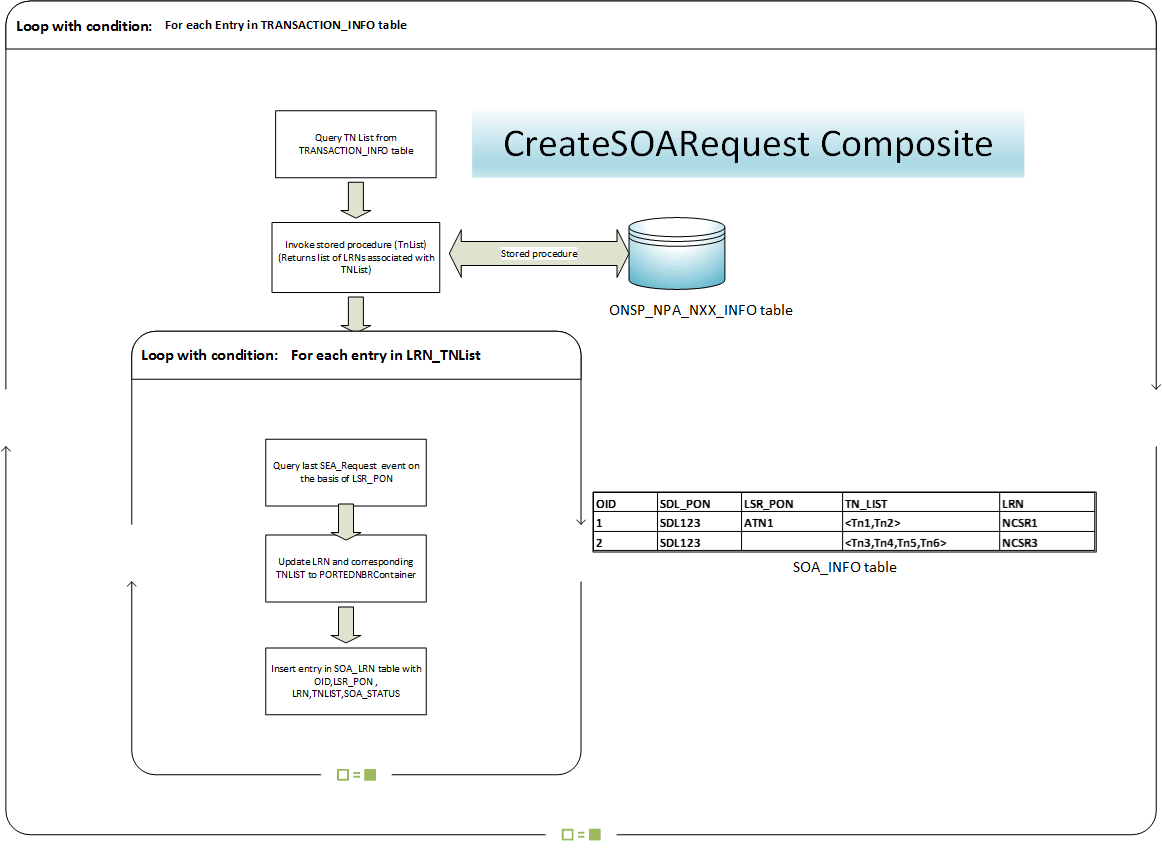
## LSR REQUEST SUBMISSION



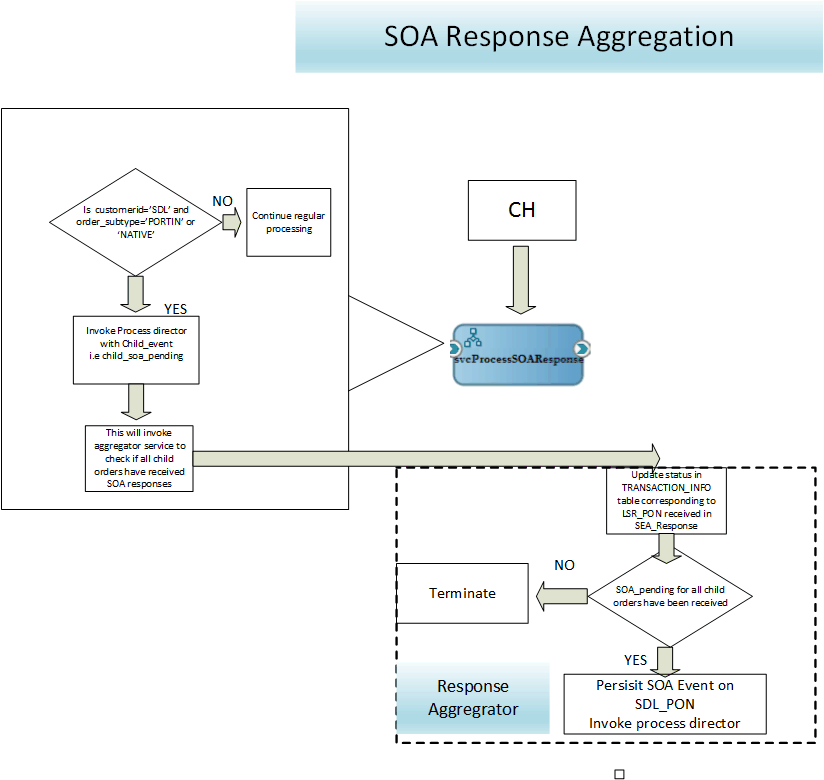
## LSR RESPONSE AGGREGRATION



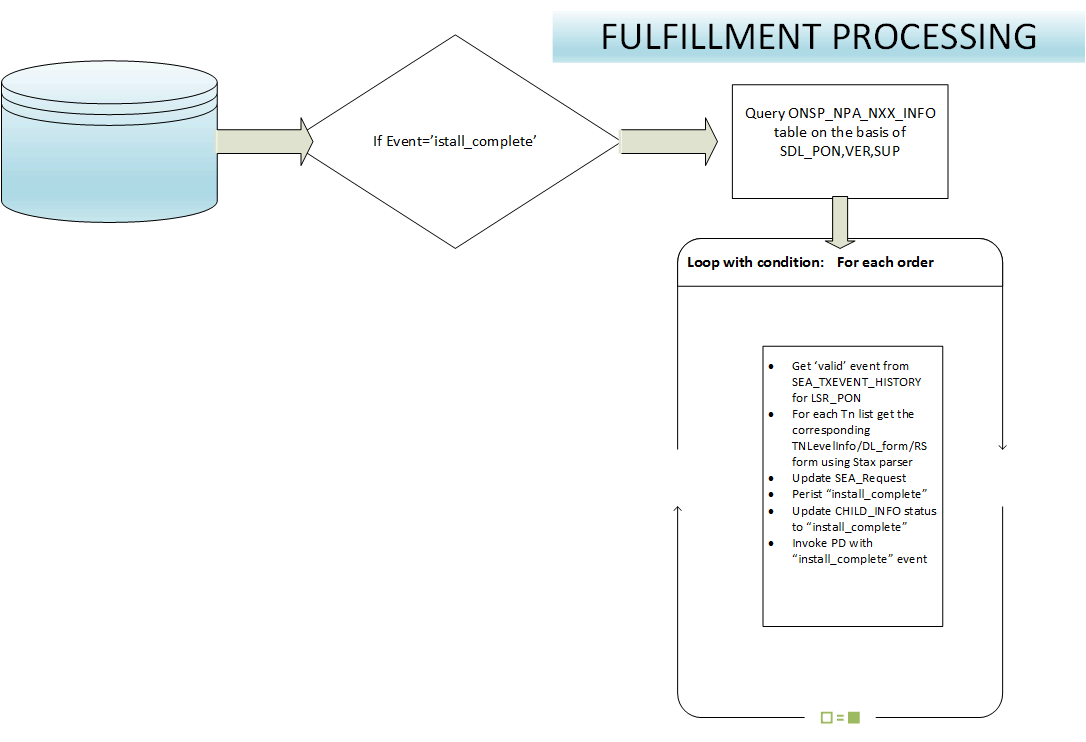
## SOA REQUEST SUBMISSION



## SOA RESPONSE AGGREGRATION



## FULFILLMENT PROCESSING



# PREVALIDATOIN AND REQUEST PROCESSING

## COMPONENT CHANGES

### MLPrevalidationUtil\_ejb – Cluster (BPEL 9.2 and BPEL 9.3 release)

Overview: - This EJB application achieves pre-validation functionalities required for SDL multiline orders. Method “doPrevalidation” defined in MLPrevalUtil Bean calls the Portps service (as jaxws client) for bulk NTS lookup, followed by Spidgateway, SIV, NpaNxx, DDF and CCNA lookup. BPEL calls the EJB bean via EJB Adapter.

Details:

For all PORTIN TN list PortPS lookup is performed

* + Map<String, ONSPInfo> onspInfoMapPortin is populated
  + For each entry in onspInfoMapPortin
    - If onsp is SDL (CCNA\_OWNER)
      * If TNs are in SeaDisco
        + Entry is added to onspInfoMapReclaim
        + Entry removed from onspInfoMapPortin
      * Else
        + Preval code with CCNA\_OWNER returned.
    - Else
      * ONSPInfo is updated with SPID Gateway and SIV info
      * DDD check is performed.
    - For list of TN in ONSPInfo NpaNxx Lookup is performed
    - Map<String, NpaNxxInfo> npaNxxInfoMapOnsp is populated and and set in ONSPInfo.
    - For each entry in npaNxxInfoMapOnsp
      * CCNA and DDF lookup is performed and entry object updated.
* for all NATIVE tn list pps lookup is performed
  + Map<String, ONSPInfo> onspInfoMapNative is populated
  + For each entry in onspInfoMapNative
    - If onsp is not SDL (CCNA\_NOTOWNER)
      * Preval code with CCNA\_NOTOWNER returned.
    - For list of TN in ONSPInfo NpaNxx Lookup is performed
    - Map<String, NpaNxxInfo> npaNxxInfoMapOnsp is populated and and set in ONSPInfo.
    - For each entry in npaNxxInfoMapOnsp
      * CCNA and DDF lookup is performed and entry object updated.
* onspInfoMapPortin, onspInfoMapReclaim, onspInfoMapNative are persisted in database in “ONSP\_NPA\_NXX\_INFO” table. “success” is returned as response indicating no preval failure.
* For SDL Green Listing Support - (BPEL 9.5.2 release)
  + If TOS=”2” and IS\_GREEN\_DIVISION from NPA-NXX lookup is “Y”
    - Populate DL\_ILEC as “GreenBook” and by-pass DDF lookup
  + Else
    - Existing logic

### BpelSynchronization – Cluster (BPEL 9.2 and BPEL 9.3 release)

Overview: - This application registers a weblogic Singleton Service in cluster, which stores lock information in a Hashmap. This service is guaranteed to be running only one instance on only one server in the cluster. This service supports Remote Method Invocation as well. Following different API calls help in maintaining synchronization across different BPEL processing flows when invoked from BPEL services via an EJB adapter.

1. canRuninCurrentState()

Input:-LockInfo (parentOid, transOid, transaction, LockTime)

Output:-boolean

This method return Boolean true, if

1. No record exists for a combination of parentOid, transOid and transaction.
2. removeOnDone()

Input:-LockInfo (parentOid, transOid, transaction, LockTime)

Output:-int

This method removes lock from store maintained based on input passed.

If parentOid, transOid and transaction passed then lock specific for all these parameter is removed.

If only parentOid is passed then locks corresponding to all transactions are removed.

### DequeueSDLRequest – Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: GENERIC\_MESSAGE\_OBJECT

Type: Synchronous

Overview: This composite will route SDL PORTIN orders to ProcessMLRequest composite.

Details:

* If order subtype is PORTIN
  + Invoke a new composite ProcessMLRequest
* else
  + Invoke svcProcessRequest. (Existing behavior for non PORTIN order subtypes)

Note: Non PORTIN Orders will be processed as they are currently processed for other customers.

### ProcessMLRequest- Enterprise (\*\*NEW\*\*) (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details:

**Note: Automation does not need transaction specific data until LSR processing**.

* Log the entire xml in ORDER\_INFO table with CLEC\_PON,DDD, VER,SUP

Note: This xml will be used in further processing to create transaction specific data. This can be stored in zip format as well.

* Create new SEA\_Request using following steps:
  + - Remove RS\_Forms/np\_forms/dl\_forms/TNLevelInfoContainer
    - This xml will be called base xml.
* The newly created xml will have minimum information that is required up to CSR transaction. This is to avoid large xml payload propagation that could slow down BPEL processing.
* Perform CH Domain lookup on the ‘new’ SEA\_Request.
* Invoke SEA\_REQUEST\_ENRICH map.
* If event=’new’
  + Persist “new” event (This event will have the base xml in SEA\_TXEVENT\_HISTORY).
  + Invoke process director with ‘new’ event. This will invoke svcSDLPreval BPEL composite.
* Else If sup in (1,2,3)
  + Invoke process director with ‘supplement’ event.

This will invoke [svcProcessSuppRequest](file:///C:\Users\amitj.yadav\AppData\Local\Microsoft\Windows\Temporary%20Internet%20Files\Content.Outlook\VQDA1K04\BPEL%209%203_Design-Multi%20LEC-ATN%20Support.docx#_SUPPlement_1_processing) with SEA\_Request

* If sup=’9’
  + Invoke process director with ‘sup3\_fulfillment’

This will invoke svcProcessSupp3FFRequest (Existing) composite with SEA\_Request.

### svcMLPortinPreval- Enterprise (\*\*NEW\*\*) (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Overview: This composite will break SDL PORTIN order into multiple child orders based on ONSP SPID/NPA-NXX and LRN combination.

* Get PORTEDNBRCONTAINER and RSNBRCONTAINER from SEA\_Request.
  + - Create a list of PORTEDNBRs and NATIVENBRs
    - Perform PortPS lookup for PORTEDNBRCONTAINER and RSNBRCONTAINER using the new PPS URL (Bulk lookup)
    - PPS service will return TN and its corresponding ONSP\_SPID as a list.
      * If PPS lookup fails
        + Go to step 10: PREVA\_CODE value=”PPS”
      * Else
        + continue
    - **Java call** : (SDLSPID, List Of <TN,ONSP\_SPID>, “PORTIN”, List of <TN,ONSP\_SPID) return by PPS for Native TNs,”NATIVE”)
      * + For “PORTIN” if any ONSP\_SPID matches with SDLSPID

Return TN and String=”PREVAL\_CODE value=”CCNA\_OWNER”

* + - * + Else

If any ONSP\_SPID doesn’t match with SDLSPID

Return TN and String=”PREVAL\_CODE value=”CCNA\_OWNER”

Else

Return success

* + - If output of java call No.3 is not “success”
      * Go to step 10.

Else

* + - * Continue.
    - Java call : (List Of <TN,ONSP\_SPID>, “PORTIN”, List of <TN,ONSP\_SPID) return by PPS for Native TNs, ”NATIVE”)
      * + Create a map ONSPSPID\_TNLIST [ONSP\_SPID,TNList] for PORTIN
        + Add all the NATIVE TNs with SDLSPID and TNList.

Example: < 0234, <Tn1,Tn2>

<6665, <Tn3,Tn4,Tn5>

<853C, <Tn6>

<513C, <Tn8,Tn7>

* + - * + Create a new Map ONSPINFO\_TNLIST <ONSPINFO,TnList>
        + For each entry in ONSPSPID\_TNLIST map

Get SPID, TnList

Create ONSP\_INFO object with (ONSP\_SPID,ONSP\_NAME,GATEWAY,SIV,BONDED\_INDICATOR,DDD\_INTERVAL,SATPORT\_INTERVAL)

Perform SPID\_Gateway lookup for the SPID

If fails

Return PREVAL\_CODE value=”SPIPD”

Update the ONSP\_INFO with information received from SPID\_GATEWAY lookup

Perform SIV\_LOOKUP lookup for the SPID

If SIV lookup fails

Return PREVAL\_CODE value=”SIV”

Update the ONSP\_INFO with information received from SIV\_LOOKUP lookup

Add ONSP\_INFO object to ONSPINFO\_TNLIST map with TNList.

Note: After we exit the above loop we would have ONSPINFO\_TNLIST() map with ONSPINFO per SPID and corresponding TNList.

* + - * + Create NPA\_NXX\_TNLIST map
        + Create ONSPINFO\_NPANXX\_TNLIST map.
        + For each entry in ONSPINFO\_TNLIST map[Break the TNs for NPA-NXX list]

For each TN in TNList

Query NPA\_NXX\_TNLIST map with the NPA\_NXX of the first TN from TNList.

Entry not found.

Add an entry in NPA\_NXX\_TNLST map with the NPA\_NXX of the TN

Add the TN to TN List.

Entry not found

Add the Tn to TN List in NPA-NXX\_TNLIST map corresponding to this NPA-NXX.

After iterating the entire TNList we will have NPA\_NXX\_TNLIST map that will contain unique NPA-NXX and TNList.

For example: NPA\_NXX\_TNLIST

<111-222, <111-222-3333,111-222-5566>

<111-333, <111-333-3434,111-333-5869>

Add ONSPINFO and NPA\_NXX\_TNLIST to ONSPINFO\_NPANXX\_TNLIST map.

ONSPINFO\_NPANXX\_TNLIST will have <ONSPINFO, NPA-NXX\_TNLIST>

* + - * + Insert the information available in ONSPINFO\_NPANXX\_TNLIST map to ONSP\_NPA\_NXX\_INFO table.

Assign OrderSubtye=’NATIVE’ if ONSP\_SPID is SDL SPID.

Else assign orderSubtype=’PORTIN’

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **OID** | **CLEC\_PON** | **ONSP\_SPID** | **GATEWAY** | **TN\_LIST** | **NPA** | **NXX** | **SIV** |
| 1 | SDL1 | A | BS | <123-123-1115;123-123-2225> | 123 | 123 | LSOG6 |
| 2 | SDL1 | A | BS | <222-222-1119;222-222-2223> | 222 | 222 | LSOG6 |
| 3 | SDL1 | B | VZE | <434-434-9999;434-434-1010> | 434 | 434 | LSOG6 |
| 4 | SDL1 | B | VZE | <103-434-9999;103-434-1010> | 103 | 103 | LSOG6 |
|  |  |  |  |  |  |  |  |

Example table data with 8 TNs and 2 ONSP’s in ONSP\_NPA\_NXX\_INFO table

* + - * + Perform DDF lookup for all unique NPA-NXX combinations available in ONSP\_NPA\_NXX\_INFO table.
        + For each ONSP\_SPID in ONSP\_NPA\_NXX\_INFO table.

For each NPA-NXX combination [NPA-NXX lookup]

Get Tn List

Get LRN info in the form of [LRN,<TNList>]

If lookup fails

Return PREVAL\_CODE value=”NPA-NXX”.

If multiple LRN found

Add existing OID to a variable “OIDToDelete”

Copy the existing row and split into as many as LRN were received also Split the TN List.

If single LRN found

Update the corresponding row with LRN value.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OID** | **CLEC\_PON** | **ONSP\_SPID** | **GATEWAY** | **TN\_LIST** | **NPA** | **NXX** | **SIV** |  | **LRN** |
| 1 | SDL1 | A | BS | <123-123-1115> | 123 | 123 | LSOG6 | K | 56555 |
| 2 | SDL1 | A | BS | <123-123-2225> | 123 | 123 | LSOG6 | K | 56588 |
| 3 | SDL1 | A | BS | <222-222-1119;222-222-2223> | 222 | 222 | LSOG6 | L | 66666 |
| 4 | SDL1 | B | VZE | <434-434-9999;434-434-1010> | 434 | 434 | LSOG6 | M | 88888 |
| 5 | SDL1 | B | VZE | <103-434-9999;103-434-1010> | 103 | 103 | LSOG6 | N | 88888 |

ONSP\_NPA\_NXX\_INFO table after LRN lookup

* + - End Java call 5:
      * If output=’success’
        + Continue.
      * Else
        + Go to step 10.
    - CCNA lookup:
      * For each gateway name
        + Perform CCNA lookup on the basis of CLEC, ILEC, and STATE combination.
        + If CCNA found

Update all rows for this CLEC,ILEC,STATE,SPID

Combinations

* + - * + If CCNA not found

Go to step 10.

* + - * + If duplicate records found for CLEC, ILEC and STATE combination.

For each NPA

Perform CCNA lookup for CLEC,ILEC,STATE and NPA combination

If found

Update corresponding row in ONSP\_NPA\_NXX\_INFO table.

If not found

Go to step 9.

If multiple records found at NPA level

For each NXX

Perform CCNA lookup for CLEC,ILEC,STATE ,NPA and NXX combination

If found

Update corresponding row in ONSP\_NPA\_NXX\_INFO table.

If not found

Go to step 9.

If multiple records found at NXX level

For each LRN

Perform CCNA lookup for CLEC,ILEC,STATE ,NPA and NXX and LRN combination

If found

Update corresponding row in ONSP\_NPA\_NXX\_INFO table.

If not found

Go to step 9.

* + - CCNA lookup ends here

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OID** | **CLEC\_PON** | **ONSP\_SPID** | **GATEWAY** | **TN\_LIST** | **NPA** | **NXX** | **SIV** | **DL\_ILEC** | **LRN** | **CCNA** | **CC** |
| 1 | SDL1 | A | BS | <123-123-1115> | 123 | 123 | LSOG6 | K | 56555 | BYH | ALY |
| 2 | SDL1 | A | BS | <123-123-2225> | 123 | 123 | LSOG6 | K | 56588 | BYH | ALY |
| 3 | SDL1 | A | BS | <222-222-1119; 222-222-2223> | 222 | 222 | LSOG6 | L | 66666 | TLB | BTW |
| 4 | SDL1 | B | VZE | <434-434-9999; 434-434-1010> | 434 | 434 | LSOG6 | M | 88888 | GTT | CBD |
| 5 | SDL1 | B | VZE | <103-434-9999; 103-434-1010> | 103 | 103 | LSOG6 | N | 88888 | GTT | CBD |

ONSP\_NPA\_NXX\_INFO table after CCNA lookup

* + - Query records from ONSP\_NPA\_NXX\_INFO table on the basis of CLEC\_PON, VER and SUP.

. If record count =1

* + - * Update SEA\_Request with data available in ONSP\_NPA\_NXX\_INFO table.
      * Persist ‘perval\_successfu’ event.

(This is for single line or single line/multi TN orders)

. Else

* + - * Persist preval\_successful event. (This will persist preval successful on original order) new->preval\_successful.

This will be a marker event on original order.

* + - * For PON, VER and SUP, query ONSP\_NPA\_NXX\_INFO table.
      * For each CC, CCNA, GATEWAY combination.

Generate CSR\_TXNUM, update CSR\_TXNUM in the table.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **CLEC\_PON** | **ONSP\_SPID** | **GATEWAY** | **TN\_LIST** | **NPA** | **NXX** | **SIV** | **DL\_ILEC** | **LRN** | **CCNA** | **CC** | **CSR\_TXNUM** |
| SDL1 | A | BS | <123-123-1115> | 123 | 123 | LSOG6 | K | 56555 | BYH | ALY | NCSR1 |
| SDL1 | A | BS | <123-123-2225> | 123 | 123 | LSOG6 | K | 56588 | BYH | ALY | NCSR1 |
| SDL1 | A | BS | <222-222-1119; 222-222-2223> | 222 | 222 | LSOG6 | L | 66666 | TLB | BTW | NCSR2 |
| SDL1 | B | VZE | <434-434-9999; 434-434-1010> | 434 | 434 | LSOG6 | M | 88888 | GTT | CBD | NCSR3 |
| SDL1 | B | VZE | <103-434-9999; 103-434-1010> | 103 | 103 | LSOG6 | N | 88888 | GTT | CBD | NCSR3 |

ONSP\_NPA\_NXX\_INFO Table

* + - * For each CSR\_TXNUM in ONSP\_NPA\_NXX\_INFO table
        + Combine TNList to create new TNList ( In example above, TN\_LIST of first and second rows will be combined)
        + Insert one entry in TRANSACTION\_INFO table per CSR\_TXNUM
        + Initially PARENT\_TXNUM will be null in TRANSACTION\_INFO table.
        + Set CSR\_COMPLETE flag to ‘Y’ for NATIVE and RECLAIM order subtype.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CLEC\_PON** | **VER** | **SUP** | **CSR\_TXNUM** | **PARENT\_TXNUM** | **CSR\_STATUS** | **CSR\_TN\_LIST** | **LSR\_PON** |
| SDL1 | 00 |  | NCSR1 | null |  | <123-123-1115; 123-123-2225> |  |
| SDL1 | 00 |  | NCSR2 | null |  | <222-222-1119; 222-222-2223> |  |
| SDL1 | 00 |  | NCSR3 | null |  | <434-434-9999; 434-434-1010; 103-434-9999; 103-434-1010> |  |

TRANSACTION\_INFO table

* + - Pre-validation failed
      * + Convert SEA\_Request to SEA\_Response.
        + Perform JCODE lookup.
        + Assign JCODE details.
        + Assign reject Message with TN list.
        + Persist ‘preval\_failed’.
        + Invoke PD.

**Notes: Step 10** **will persist “preval\_failed” on the master order and order will be returned back to SDL adapter.**

* + - Invoke PD with preval\_successful event (SEA\_Request) this will invoke CreateCSRRequest composite.

### GenerateCSR – Enterprise (\*\*NEW\*\*) (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Overview:

* + - * If ActivityType=”RedoCSR”
  + If ONSP\_NAME is in “ secondCSRIlec” dd property and csrInqIndicator is not equal to “A” in transaction info
    - Generate new TXNUM and update in ONSP\_NPA\_NXX\_INFO and Transaction\_info tables
    - Update CSR\_INQ\_TYPE = “A” in Transaction\_Info table
    - Query CC,CCNA from ONSP\_NPA\_NXX\_INFO table
  + Else
    - Create new record in ONSP\_NPA\_NXX\_INFO and Transaction\_Info table corresponding to input CSRTXNUM with updated TXNUM value.
    - Update new Transaction\_Info record with ATN as input ATN, CSR\_COMPLETE=”Y”, CSR\_STATUS=”csr\_successful”, MI\_IND, csr\_tn\_list as populated from input.
    - Query CC,CCNA from ONSP\_NPA\_NXX\_INFO table
      * Else
  + For each entry in TRANSACTION\_INFO table for SDL\_PON, VER and SUP and order\_subtype=’PORTIN’
    - * Query CC,CCNA from ONSP\_NPA\_NXX\_INFO table
      * Update in SEA\_Request with CC, CCNA, TXNUM, BONDED\_INDICATOR, ONSP\_NAME etc.
      * Assign first TN from CSR\_TN\_LIST to PORTEDNBRContainer of SEA\_Request
      * Assign first TN to ATN of SEA\_Request.
      * Remove rest of the PORTEDNBRs from PORTEDNBRContainer.
      * Invoke svcCSRProcessing (Existing) BPEL service.
      * Note: This will invoke svcCSRProcessing🡪 svcCSRRequest🡪svcCHAsyncRequest.

TXNUM is already generated hence BPEL maps will be updated to bypass TXNUM generation.

After we finish the loop, CSR will be submitted for each entry in TRANSACTION\_INFO table.

### svcCSRProcessing – ENTERPRISE (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Overview:

* If ONSP\_NAME is not in bonded ILEC DD property
  + - If MixONSPCustomers DVM property contains customerName
      * Update TRANSACTION\_INFO table corresponding to incoming TXNUM and set CSR\_COMPLETE=’YES’;
      * Invoke PD with “ml\_csr\_successful” event. (Existing behavior).
* Submit CSR Request as per current functionality.
  + If MixONSPCustomers DVM property contains customerName
    - Set stale check flag false.
    - Perform stale check(existing functionality)
    - If csr\_response is submitted within stale hours
* For SDL set stale check true if csr\_response is submitted within stale hours.
* Update csr\_txnum column in TRANSACTION\_INFO table with old txnum of order.
  + - Else If CSR rejected
      * Bypass persisting csr\_rejected in SEA\_TXEVENT\_HISTORY.
      * Update CSR\_XML column in TRANSACTION\_INFO table corresponding to TXNUM.( This column will have reject xml for redo from GUI)
      * Invoke PD with “ml\_csr\_rejected”. (This will invoke ResponseAgregator service).
    - if CSR submitted
      * Bypass svcProcessCSRRequet service invocation. (Do not persist csr\_submitted on child order).
    - If stale check flag is true
      * Invoke svcSDLPreval service.

### svcRouteCSRResponse –Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_CSRResponse

Type: Asynchronous

* If MixONSPCustomers DVM property contains customerName and TXNUM is present in TRANSACTION\_INFO table.
  + - Update event to “ml\_csr\_response”.
    - Invoke svcProcessCSRResponse.

### svcProcessCSRResponse- Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_CSRResponse

Type: Asynchronous

* If event=’ml\_csr\_response’
  + - Get CLEC\_PON from TRANSACTION\_INFO table based on TXNUM.
    - Get trans record from SEA\_ESR\_TRANS table based on CLEC\_PON and CustomerName
    - If lastResponseOID does not exist
      * Get last SEA\_Request on the basis of lastRequsetOID.
      * Convert SEA\_Request to SEA\_Response
    - Copy CSR\_Respnse data from SEA\_CSRResponse to SEA\_Response.
    - Perform regular CSR processing.
    - If TNMatch jcode processing returns fault (TN not match)
      * Assign csr data in SEARequest
      * Invoke GenerateCSR with activityType=”RedoCSR”
    - If MixONSPCustomers DVM property contains customerName
      * Bypass persisting the event into SEA Database.
    - Update response status (csr\_successful/csr\_failure) in TRANSACTION\_INFO table based on TXNUM.
    - Invoke PD with ml\_csr\_successful/ml\_csr\_failure event

Note: PD call on ml\_csr\_successful/ml\_csr\_failure event will invoke a new composite MLResponseAggregator

### svcProcessCSRRequest – Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_CSRRequest

Type: Asynchronous

* If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’
  + Updated csr\_status to csr\_submitted in transaction\_info table
* If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’
  + Prefix ml\_ in event.

### svcSDLEnqueueResponse – Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Response

Type: Asynchronous

* If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’
  + Updated csr\_status to csr\_submitted in transaction\_info table
* If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’
  + Prefix ml\_ in event.

### MLResponseAggregator: Enterprise (\*\*NEW\*\*) (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Response

Type: Asynchronous

Details: This composite will have following DVM properties.

**NegativeEvents:** csr\_failure, order\_returned, csr\_rejected, lsr\_rejected, jeopardy, focreject.

**PositiveEvents**:

csr\_successful, focaccept, soa\_pending, soa\_modify\_successful, soa\_rel\_successful, completed, cancelled

* If IncomingEvent in (‘csr\_failure’,’csr\_rejected’,)
  + - * Query TRANSACTION\_INFO table based on TXNUM
      * Query TRANSACTION\_INFO table on the basis of SDL\_PON.
        + If incoming event is not present in TRANSACTION\_INFO table for SDL\_PON.

Update TRANSACTION\_INFO table set CSR\_STATUS=incomingEvent.

If SDL\_PON!= SEA\_Response/PON

Query SEA\_Response on the basis of SDL\_PON from SEA\_TXEVENT\_HISTORY

If found

Continue.

Not found

Query SEA\_Request on the basis of SDL\_PON and convert it to SEA\_Response.

Update error in SEA\_Response.

invoke canRunInCurrentState operation of synchronizationManager service with parent\_oid, trans\_oid and transaction

If return true

Persist incoming event.

Invoke process director with incoming event

Else

Continue

Else

Continue

* + - * + Else

Update TRANSACTION\_INFO table set CSR\_STATUS=incomingEvent.

* Else If IncomingEvent available in NegativeEvents
  + - * Query TRANSACTION\_INFO table based on LSR\_PON
      * Get SDL PON and Query TRANSACTION\_INFO table on the basis of SDL\_PON.
        + If incoming event is not present in TRANSACTION\_INFO table for SDL\_PON.

Update TRANSACTION\_INFO table set LSR\_STATUS=incomingEvent.

If SDL\_PON!= SEA\_Response/PON

Query SEA\_Response on the basis of SDL\_PON from SEA\_TXEVENT\_HISTORY

If found

Continue.

Not found

Query SEA\_Request on the basis of SDL\_PON and convert it to SEA\_Response.

Update error in SEA\_Response.

invoke canRunInCurrentState operation of synchronizationManager service with parent\_oid, trans\_oid and transaction

If return true

Persist incoming event.

Invoke process director with incoming event

Else

Continue

else

Continue.

* + - * + Else

Update TRANSACTION\_INFO table set CSR\_STATUS=incomingEvent.

* If IncomingEvent in (‘focaccept’,’valid’)
  + - * Query TRANSACTION\_INFO table based on LSR\_PON
      * Get SDL PON and Query TRANSACTION\_INFO table on the basis of SDL\_PON.
      * Update LSR\_STATUS=incomingEvent
      * If count(\*) in TRANSACTION\_INFO table for the SDL\_PON = count(\*) for event=’focaccept’ and order\_subtype=’PORTIN’ + count(\*) for event=’valid’ and order\_subtype=’NATIVE’

If SDL\_PON!= SEA\_Response/PON

Query SEA\_Response on the basis of SDL\_PON from SEA\_TXEVENT\_HISTORY

If found

Continue.

Not found

Query SEA\_Request on the basis of SDL\_PON and convert it to SEA\_Response.

Update error in SEA\_Response.

invoke canRunInCurrentState operation of synchronizationManager service with parent\_oid, trans\_oid and transaction

If return true

Persist incoming event.

Invoke process director with incoming event

Else

Continue

else

Continue.

* + - * Else
        + Terminate.
* Else if IncomingEvent available in positiveEvents and event is not ‘csr\_successful’
  + - * Query TRANSACTION\_INFO table based on LSR\_PON
      * Get SDL PON and Query TRANSACTION\_INFO table on the basis of SDL\_PON and incomingEvent.
      * Update LSR\_STATUS=IncomingEvent
      * If all rows corresponding to SDL\_PON have status=incomingEvent.

If SDL\_PON!= SEA\_Response/PON

Update SDL\_PON in SEA\_Response.

Invoke synchronizationManager EJB service on canRunInCurrentState operation.

If synchronizationManager service return true

* Persist incoming event.

Else

* Continue

Else

Update SDL\_PON in SEA\_Response

invoke canRunInCurrentState operation of synchronizationManager service with parent\_oid, trans\_oid and transaction

If return true

Persist incoming event.

Invoke process director with incoming event

Else

Continue

* Else if IncomingEvent is ‘csr\_successful’
  + - Query TRANSACTION\_INFO table based on TXNUM received in SEA\_Response.
    - If ONSP\_NAME (‘ATT’,’QWEST’,’SPRINT)
      * If WTNContainer has a single TN.
        + Update the ATN received in CSR response in TRANSACTION\_INFO table.
        + Generate new TXNUM and update CSR\_TXNUM column in the existing record.
        + Query ‘new’ SEA\_Request from SEA\_TXEVENT\_HISTORY.
        + Query CC,CCNA from ONSP\_NPA\_NXX\_INFO table
        + Update in SEA\_Request with CC, CCNA, TXNUM, BONDED\_INDICATOR, ONSP\_NAME etc.
        + Assign ATN from TRANSACTION\_INFO table to ATN and First TN from TNList to PORTEDNBRContainer of SEA\_Request
        + Remove rest of the PORTEDNBR from PORTEDNBRContainer.
        + Invoke svcCSRProcessing (Existing) BPEL service.

Note: This will invoke svcCSRProcessing🡪 svcCSRRequest🡪svcCHAsyncRequest and CSR for this TXNUM will be submitted.

* + - * + Terminate
      * Else
        + Continue.
    - Remove TN from TNList based on CSR response WTNContainer, assign leftover TNs to newTNList variable.
    - Update the current record with ATN, updated TnList. And status as csr\_successful.
      * If newTnList is not empty
        + Copy existing record
        + Assign CSR\_TXNUM to PARENT\_TXNUM
        + Generate new TXNUM and assign it to CSR\_TXNUM
        + Assign newTNList to CSR\_TN\_LIST
        + Insert the new record.
        + Query ‘new’ SEA\_Request from SEA\_TXEVENT\_HISTORY.
        + Query CC,CCNA from ONSP\_NPA\_NXX\_INFO table
        + Update in SEA\_Request with CC, CCNA, TXNUM, BONDED\_INDICATOR, ONSP\_NAME etc.
        + Assign first TN from CSR\_TN\_LIST to PORTEDNBRContainer of SEA\_Request
        + Remove rest of the PORTEDNBR from PORTEDNBRContainer.
        + Invoke svcCSRProcessing (Existing) BPEL service.
        + Note: This will invoke svcCSRProcessing🡪 svcCSRRequest🡪svcCHAsyncRequest and CSR for this TXNUM will be submitted.
      * If newTnList empty
        + Mark PARENT\_TXNUM hierarchy to CSR\_COMPLETE=’Y’

Back trance each record on the basis of PARENT\_TXNUM until PARENT\_TXNUM is found to be null.

Set CSR\_COMPLETE=’Y’ for each record found.

* + - * If all entries in TRANSACTION\_INFO table corresponding to CLEC\_PON have CSR\_COMPLETE=’Y’ {This means that the CSR activity for this SDL PON has ended we can proceed creating LSR transactions}
      * invoke canRunInCurrentState operation of synchronizationManager service with parent\_oid, trans\_oid and transaction
        + If return true

Persist csr\_successful {This will persist csr\_succesful on master record}

Invoke PD with create\_child event.

This will invoke CreateChildOrder BPEL composite.

* + - * + Else

Continue

* + - * else

Terminate

### CreateChildOrders (\*\*NEW\*\*) (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Response

Type: Asynchronous

Details: This composite will create child orders.

If event=’csr\_successful’

* + For each entry in TRANSACTION\_INFO table.
    - Query SEA\_ESR\_TRANS on the basis of SDL PON available in incoming SEA\_Response.
    - Query ‘new ’ (SEA\_Request) event from SEA\_TXEVENT\_HISTORY (This will provide base xml)
    - Query TRANSACTION\_INFO table on the basis of CLEC\_PON, VER and SUP.
      * If record count=1 (Single TN or multi TN single ONSP order)

(This means that no child’s were generated and we have persisted preval\_successful with validated data on SDL\_PON)

* + - * + Do not generate LSR PON.
        + Update LSRPON from SEA\_Request in TRANSACTION\_INFO table.
        + If order\_subtype=’PORTIN'

Invoke process director with csr\_successful event.

* + - * + If order\_subtype=’NATIVE’

Invoke process director with ‘preval\_successful’ event.

* + - * + Terminate.
      * Else
        + Generate LSRPON and update in SEA\_ Request.
        + Update PORTEDNBRContainer with CSR\_TN\_LIST from TRANSACTION\_INFO table.
        + Update LSRPON in TRANSACTION\_INFO table.
        + Update MML\_INDICATOR to ‘S’ for this lsrpon trans
    - Update transaction specific data.
      * Query original XML\_DATA from ORDER\_INFO table based on SDL PON, VER and SUP indicator.
      * Query XML\_DATA for data specific to validate request.
      * Update SEA\_Request with information retrieved from original XML\_DATA.
      * Update ATN from the transaction info table record
      * Persist ‘new’.
      * If not a single line ml order
        + Update mml\_indicator to ‘M’ for clecpon trans.
      * Update SEA\_ESR\_TRANS table set MML\_INDICATOR field as ‘S’ for each child order and ‘M’ for Master Order.
      * If order\_subtype=’PORTIN’
        + Invoke process director with ‘csr\_successful’ event.
      * If order\_subtype=’NATIVE’
        + Invoke process director with ‘preval\_successful’ event.

### svcMLReturnChildOrders (\*\*NEW\*\*) (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Response

Type: Asynchronous

Overview: This composite returns the master order and persist order\_returned on master and it’s corresponding child orders.

* Query TRANSACTION\_INFO table on the basis of CLEC\_PON and VER available in incoming SEA\_Response.
  + - * If record count=1 (Single TN or multi TN single ONSP order)

(This means that either no child was generated or single line order)

* + - * Get trans record from SEA\_ESR\_TRANS table based on CLEC\_PON,VER and CustomerName.
      * If lastResponseOID does not exist
      * Get last SEA\_Request on the basis of lastRequestOID.
      * Convert SEA\_Request to SEA\_Response
      * Copy RejectMessage, Remarks, RejectCodesContainer and Customer\_Ticket\_id from incoming SEA\_Response to new SEA\_Response.
      * Invoke PD with return\_order event.This will invoke svcReturnOrder Composite.
      * Else
      * For each record get trans record from SEA\_ESR\_TRANS table based on its LSR\_PON, VER and CustomerName.
      * If lastResponseOID does not exist
      * Get last SEA\_Request on the basis of lastRequestOID.
      * Convert SEA\_Request to SEA\_Response
      * Copy RejectMessage, Remarks, RejectCodesContainer and Customer\_Ticket\_id from incoming SEA\_Response to new SEA\_Response.
      * For each order Persist order\_returned on all child orders.
      * For CLEC\_PON invoke PD with return\_order event. This will invoke svcReturnOrder.

### svcCreateValid – Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details: This composite will create valid on individual child orders.

* + This composite receives SEA\_Request.
  + If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’
    - Query CSR\_Response xml from SEA\_PRE\_LSR\_TRANS (This is for CSR import functionality)
  + If MixONSPCustomers DVM property contains customerName and order\_subtype=’NATIVE’
    - Invoke with ml\_valid event. This will invoke MLResponseAgregrator composite.

### svcProcessLSRResponse – Enterprise (BPEL 9.4 release)

Input: SEA\_Response

Type: Asynchronous

Details:

* if DL success response and SupplierName not ‘CCST’
  + If DLIlec = ‘ATTSE’
    - continue
  + else
    - get DL\_MIGRATION\_CONFIG corresponding to customername, suppliername, ordertype, ordersubtype
    - if dl\_migration\_config exist
      * if first dl submission for trans
        + get dl\_submitted event
        + if DL\_MIGRATION\_INDICATOR is D in dl\_submitted

persist response and invoke dlSubmit with DL\_MIGRATION\_INDICATOR as ‘N’

done.

* + - * + Else

Continue.

* + - * Else
        + continue.
  + else
    - continue
* populate lsr\_adminsection PON from sequence and DL\_MIGRATION\_INDICATOR
* invoke updateDDD and populate DDD from response nextDate\_noWait element.
* invoke svcLSRRequest.
* If MixONSPCustomers DVM property contains customerName
* Invoke PD with ml\_event event. Where event could be focaccept,focreject,jeopardy etc.

This will call to MLResposeAggregator service.

### svcSOASubmit –Enterprise (BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details: Service updated to assing accountID and accountName in SvCreateRequest for Mix Onsp Customer.

* + If MixONSPCustomers and SvCreateRequest
    - Get TransactionInfo for pon and ver
    - Get OrderInfo for clecPon, ver
    - Assign OrderInfo ATN field to SvCreateRequest’s accountName field and 999-<SOA\_ACCOUNTID\_SEQ.nextval()> to SvCreateRequest’s accountID

### svcProcessSOAResponse –Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Response

Type: Asynchronous

Details:

* + If MixONSPCustomers DVM property contains customerName
    - Invoke PD with ml\_event event. Where event could be soa\_pending,soa\_modify\_successful etc.

This will call to MLResposeAggregator service.

### ANInstallCompleteRequest - Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details:

This service will be invoked from dequeueSDLRequest with SEA\_Request

* + If MixONSPCustomers DVM property contains customerName
    - Query child order from TRANSACTION\_INFO on the basis of SDL PON,VER,CustomerID.
    - For each child order
      * Get ‘valid’ or ‘fulfillment\_submitted’ from SEA\_TXEVENT\_HISTORY
      * For each PORTEDNBRContainer in SEA\_Request
        + Get TNLevelInfoContainer/dl\_forms/RS\_Forms and update in SEA\_Request
      * Persist install\_complete and invoke process director.

Fulfillment will be triggered here.

* + - Persist fulfillment\_submitted on parent order.

### svcOrderComplete – Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details:

* + If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’ or ‘NATIVE’
    - Invoke process director with ml\_completed event. (This will invoke MLResponseAgregator service)

### svcCancelOrder – Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details:

* + If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’ or ‘NATIVE’
    - Invoke process director with ml\_cancelled event. (This will invoke MLResponseAgregator service)

### PurgeMLTransRecord – Enterprise (New) (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Response

Type: Asynchronous

Details: This composite will be invoked on cancelled, completed event for PORTIN order subtype.

* + - Delete data from ONSP\_NPA\_NXX\_INFO/TRANSACTION\_INFO/SOA\_INFO table on the basis of SEA\_Resposne/PON.
    - Invoke removeOnDone operation of the BpelSynchronization service. This will remove all locks from store corresponding to parent\_oid.

### svcDLIProcessing – Enterprise (BPEL 9.4 release)

Input: SEA\_Request

Type: Asynchronous

Details: Composite updated to submit DLI if configured in DL\_MIGRATION\_FONFIG Table except for CCST.

* If DL\_MIGRATION\_CONFIG record found corresponding to customerName, supplierName, orderType, orderSubType
  + Submit DLI request
* Else
  + Invoke PD with dli\_successful.

### svcProcessDLIResponse – Enterprise (BPEL 9.4 release)

Input: SEA\_DLIResponse

Type: Asynchronous

Details: This composite will be invoked on DLI response.

* If DLI Success Response and SupplierName=’CCST’
  + Existing processing
* Else
  + Invoke PD with response event.

### svcProcessPendingTransaction – Enterprise (BPEL 9.4 release)

Input: string

Type: Asynchronous

Details: Composite updated to process pending LIDB and DL transactions.

* If varTransactionType = 'DL'
  + Assing activityType to ‘submit\_dl’
  + Invoke svcDLSubmit

### svcDLSubmit – Enterprise (BPEL 9.4, 9.5 release)

Input: SEA\_Request

Type: Asynchronous

Details: This composite will be invoked to submit DL.

* If DL\_ILEC = '' or CustomerName='SDL'
  + Perform NpaNxx Lookup and assign DLIlec and Lex in request from NpaNxx result.

-------------------

* If ActivityType = ‘submit\_DL’
  + Continue
* Else
  + updateDDD Calculation
  + if no diff in inDate and calculated nextDate
    - continue
  + else
    - persist DL Pending
    - insert record in SEA\_TRANSACTION\_HOLDING
    - throw DoneFault
* If DLIlec = ‘ATTSE’
  + continue
* else
  + get DL\_MIGRATION\_CONFIG corresponding to customername, suppliername, ordertype, ordersubtype
  + if dl\_migration\_config exist
    - if first dl submission for trans
      * get dli\_successful event
      * Calculate TOS, TOA diff and update DL\_MIGRATION\_INDICATOR accordingly.
    - Else
      * Check for DL\_MIGRATION\_INDICATOR and if not exist populate from dl\_pending.
  + else
    - continue
* populate lsr\_adminsection PON from sequence and DL\_MIGRATION\_INDICATOR
* invoke updateDDD and populate DDD from response nextDate\_noWait element.
* invoke svcLSRRequest.

### svcProcessSuppRequest – Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details:

* + This composite receives SEA\_Request.
  + This composite will perform checks on supplement request.
    - If SUP is not acceptable
      * Order return supplement to SDL.
    - Else
      * If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’
        + Invoke svcProcessMLSuppRequest via Process director.

### ProcessMLSupplementRequest - Enterprise (BPEL 9.2 and BPEL 9.3 release)

Input: SEA\_Request

Type: Asynchronous

Details:

This service will receive SEA\_Request

* + Query TRANSACTION\_INFO table on the basis of CLEC\_PON and VER not the incoming VER
  + If LSR\_PON is null in TRANSACTION\_INFO (This means that child orders were not created)
    - Invoke process director with send\_<supplementType> event witih incoming SEA\_Request
  + If LSR\_PON is not null and SUP=1 or SUP=2
    - For each entry in TRANSACTION\_INFO table.
      * Get trans record on the basis of LSR\_PON
      * Get last request from SEA\_TXEVENT\_HISTORY
      * Update incoming VER,SUP,DDD in SEA\_Request
      * Invoke PD with send\_<SupplementType> event.
  + If LSR\_PON is not SUP=3
    - Invoke process director with incoming SEA\_Request and event=’send\_supplement3’
    - For each entry in TRANSACTION\_INFO table.
      * Get trans record on the basis of LSR\_PON
      * Get last request from SEA\_TXEVENT\_HISTORY
      * Invoke svcCancelSOA bpel composite.

### svcLSRSubmit – Enterprise (BPEL 9.5.2 release)

Input: SEA\_Request

Type: Asynchronous

Details: - service will be updated to remove DL from in case of GreenBook

* + if Customer name is in Mix ONSP customer (from DD Property) and DL\_ILEC = “GreenBook”
    - Remove All DL Forms.
  + Else
    - ELT/LTY lookup will happen and existing flow would resume.

### svcUpdateDDD – Enterprise (BPEL 9.5 release)

Input: svcUpdateDDDProcessRequest

Type: Synchronous

Details: - This service perform Date calculation based on SEA\_ILEC\_CONFIG table data.

* + isDL\_ilec = 'DL
    - assign DelayDLHours from SEA\_ILEC\_CONFIG table to initialWaitTime for Date next date calculation.
    - Perform Date calculation as existing logic.

# Installation Impact

Installation guide will capture all the steps to deploy the changes in production.

# Product Impact

## ESR GUI

Automation will not persist transaction history in ORDER/TRANS/History tables until CSR processing for all child order is complete. This is to avoid inconsistent data logging as CSR lookup may split the order into multiple orders based on ATN value.

After CSR transactions, automation will start persisting events in SEA database tables.

ESR GUI will require changes to provide re-do mechanism for CSR transactions from new database tables mentioned in database changes section.

## REPORTIN and BILLING

Reporting and billing scripts needs to be modified due to new database tables introduced for CSR transaction logging.

# Data Model Changes

## SEQUENCE

## SCHEMA CHANGES –SEA SCHEMA

### ORDER\_INFO

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Remarks** |
| OID | NUMBER (38)  Primary Key |  |
| CLEC\_PON | VARCHAR2 (32Byte)  Primary Key | This will hold the actual PON that Suddenlink has sent in the initial request |
| VER | VARCHAR2 (32Byte)  Primary Key | Version of the SDL request |
| SUP | VARCHAR2(5 BYTE) | SUP indicator |
| DDD | TIMESTAMP (0) | Due date requested by Suddenlink. |
| XML\_DATA | CLOB | Original request xml received from adapter. |
| ATN | VARCHAR2(22) | SDL ATN received in request. |
| CREATED\_DT | TIMESTAMP(6) |  |

### ONSP\_NPA\_NXX\_INFO

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Remarks** |
| OID | NUMBER (38)  Primary Key |  |
| CLEC\_PON | VARCHAR2 (32Byte)  Primary Key | This will hold the actual PON that Suddenlink has sent in the initial request |
| VER | VARCHAR2 (32Byte)  Primary Key | Version of the SDL request |
| SUP | VARCHAR2(5 BYTE) | SUP indicator |
| ONSP\_SPID | VARCHAR2(5 BYTE) | . |
| GATEWAY | VARCHAR2(200 BYTE) |  |
| TN\_LIST | VARCHAR2(200), |  |
| NPA | VARCHAR2(3) |  |
| NXX | VARCHAR2(3) |  |
| NPA\_NXX\_LRN | VARCHAR2(12 BYTE) |  |
| DL\_ILEC | VARCHAR2(200) |  |
| LEX | VARCHAR2(8 BYTE), |  |
| LRN | VARCHAR2(12 BYTE) |  |
| SIV | VARCHAR2(200) |  |
| BAN | VARCHAR2(20 BYTE) |  |
| CIC | VARCHAR2(20 BYTE) |  |
| CC | CHAR(4 BYTE) |  |
| CCNA | CHAR(4 BYTE) |  |
| JB\_CCNA | VARCHAR2(4 BYTE) |  |
| JB\_CC | VARCHAR2(4 BYTE) |  |
| DDD\_INTERVAL | NUMBER(3,0) |  |
| BONDED\_INDICATOR | CHAR(1 BYTE) |  |
| SATPORT\_INTERVAL | VARCHAR2(2 BYTE) |  |
| STATE | VARCHAR2(3 BYTE) |  |
| CSR\_TXNUM | VARCHAR2(16 BYTE) |  |
| ORDER\_SUBTYPE | VARCHAR2(32 BYTE) |  |
| CREATED\_DT | TIMESTAMP(6) |  |

### TRANSACTION\_INFO

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Remarks** |
| OID | NUMBER (38)  Primary Key |  |
| CLEC\_PON | VARCHAR2 (32Byte)  Primary Key | This will hold the actual PON that Suddenlink has sent in the initial request |
| VER | VARCHAR2 (32Byte)  Primary Key | Version of the SDL request |
| SUP | VARCHAR2(5 BYTE) | SUP indicator |
| ATN | VARCHAR2(22 BYTE) | . |
| CSR\_TXNUM | VARCHAR2(16 BYTE) |  |
| PARENT\_TXNUM | VARCHAR2(16 BYTE) |  |
| CSR\_STATUS | VARCHAR2(22 BYTE) |  |
| CSR\_TN\_LIST | VARCHAR2(200) |  |
| LSR\_PON | VARCHAR2(32 BYTE) |  |
| LSR\_STATUS | VARCHAR2(22 BYTE) |  |
| ORDER\_STATUS | VARCHAR2(22 BYTE) |  |
| CSR\_INQ\_TYPE | VARCHAR2(3) |  |
| MI\_IND | VARCHAR2(3) |  |
| CREATED\_DT | TIMESTAMP(6) |  |

### SOA\_INFO

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Remarks** |
| OID | NUMBER (38)  Primary Key |  |
| CLEC\_PON | VARCHAR2 (32Byte)  Primary Key | This will hold the actual PON that Suddenlink has sent in the initial request |
| LSR\_PON | VARCHAR2 (32Byte) |  |
| SOA\_TN\_LIST | VARCHAR2(200) |  |
| LRN | VARCHAR2(12 BYTE) | . |
| SOA\_STATUS | VARCHAR2(22 BYTE) |  |
| ORDER\_SUBTYPE | VARCHAR2(22 BYTE) |  |
| HISTORY\_OID | NUMBER(38) |  |
| CREATED\_DT | TIMESTAMP(6) |  |

### SEA\_DL\_MIGRATION\_CONFIG

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Remarks** |
| CUSTOMER | VARCHAR2(20)  Primary Key |  |
| SUPPLIER | VARCHAR2(20)  Primary Key |  |
| ORDER\_TYPE | VARCHAR2 (32) |  |
| ORDER\_SUBTYPE | VARCHAR2 (32) |  |

### sea\_ilec\_config

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Type** | **Remarks** |
| DELAY\_LIDB\_HOURS | NUMBER | Column added to hold delay in hours for lidb submission. |
| DELAY\_DL\_HOURS | NUMBER | Column added to hold delay in hours for DL submission. |

## Oracle JOB/Schedule Configuration

## STORED PROCEDURE (SEA DB)

# Component Changes

## Schema CHANGES

SEA\_Request.xsd

SEA\_Base.xsd

SEA\_Header xsd

## XSL transformation Templates

## Process Director

## JCODE CHANGES

# Re-do functionality form ESR GUI

# TIER 2 REFLOW SERVICES

Automation will not log CSR transactions in regular transaction tables hence new reflow services will be provided to reflow such orders.

### clearSynchronizationLock (Tier2Services): Lock present in synchronization store and will not allow reflow to complete correctly (BPEL 9.2 release)

Input: parentOid, transOid, transaction

Output: true/false

Type: Synchronous

This service will remove lock from store maintained based on input passed. This should be invoked before reflow if in case reflow is not working due to lock present in the store. If parentOid, transOid and transaction passed then lock specific for all these parameter is removed.

If only parentOid is passed then locks corresponding to all transactions are removed.

## ORDER stuck in PREVAL SUCCESSFUL EVENT

Pre-condition: Suddenlink PORTIN order stuck in ‘preval\_successful’ event and there is no record in TRANSACTION\_INFO table corresponding to CLEC\_PON and VER.

Steps to reflow:

* Invoke below new BPEL composite “GenerateTransInfo” with SEA\_Request of preval\_successful event.

### GenerateTransInfo (Tier2Services): Order stuck in preval\_successful (BPEL 9.2 release)

Input: SEA\_Request

Type: Asynchronous

Details:

* Invoke GENERATE\_TRANSACTION\_INFO stored procedure to generate TRANSACTION\_INFO records on the basis of CLEC\_PON and VER.
* Invoke GenerateCSR BPEL composite with incoming SEA\_Request.

## ORDER stuck in CSR\_REJECTED/CSR\_FAILURE event

Pre-condition: CSR\_STATUS corresponding to TXNUM in TRANSACTIO\_INFO table is csr\_rejected or csr\_failure.

### SubmitCSRRequest (Tier2Services): Order stuck in preval\_successful (BPEL 9.2 release)

Input:

clecPon:- String

customerId:-String

txNumCsvList:-Comma separated list of TXNUM String

bypassCSR:- String

Type: Asynchronous

Details:

* If bypassCSR flag = y
  + Get last request from SEA\_TXEVENT\_HISTORY
  + Convert SEARequest to SEAResponse
  + Updated CSR\_STATUS = “csr\_successful” and CSR\_COMPLETE=”Y” in transaction\_info table corresponding to csr\_txnum in txNumCsvList.
  + Update SEAResponse txnum element with first txnum from txNumCsvList and event to “csr\_successful”
  + Invoke pd with “ml\_csr\_successful”
* else
  + Query “preval\_successful” SEA\_Request from SEA\_TXEVENT\_HISTORY.

For Each TXNUM

* + Create new TXNUM and update in TRANSACTION\_INFO and ONSP\_NPA\_NXX\_INFO.
  + Get record corresponding to new txnum and Create CSR\_Request,

Invoke svcCSRProcessing service.

## order return mix onsp order

### svcMLReturnOrder (Tier2Services): (BPEL 9.2 release)

Input:

clecPon:-String

ver:-String

customerId:-String

jCode:-String

rejectMessage:-String

remark:-String

Type: Asynchronous

Details:-

* for all child pon(lsr\_pon) in transaction\_info table corresponding to clecpon,ver
  + get SEAResponse for lsr\_pon.
  + Updated jcode, rejectMessage, remark if provided.
  + Update event to “order\_returned”
  + Persist response.
* Get SEAResponse for clecpon
* Update jcode, rejectMessage, remark if provided
* Invoke ProcessDirector with event return\_order.

## CHANGES to existing BPEL composite

Following BPEL composites will be updated for reflowing events for SDL customer.

### svcProcessSEAResponse (Tier2Services): (BPEL 9.2 release)

Input: SEA\_Response

Type: Asynchronous

* If MixONSPCustomers DVM property contains customerName and order\_subtype=’PORTIN’
  + Persist incoming event.
  + Invoke processDirector with event in input SEA\_Response prefixed with “ml\_”.

## ASSUMPTIONS

|  |
| --- |
| **Assumption** |
|  |

# SCENARIOS TO BE CONSIDERED

|  |
| --- |
| Fallout handling for CSR transaction is not converted. |
| Partial SOA failures are will not be processed correctly. |

## Glossary of Terms

|  |  |  |
| --- | --- | --- |
| **Term / Phrase** | **Explanation** | |
|  | |  |

Table – Glossary of Terms