Document Attributes

| Attribute | Value |
| --- | --- |
| Application ID /  Application Name | MOTS ID:  14724 / GCP (Global Computing Platform) |
| Owner | GCP: Ming Ho |
| Owner Contact Information | mh6892, 732-420-7673 |
| Other Attribute | Project ID:  289116.141314  270843d  289207 |

## 

Revision History

The following table lists the revision history of this document:

| Author | Date | Version # | Revision Description |
| --- | --- | --- | --- |
| Ming Ho | 3/25/2016 | 0.01 | <289116.141314> Initial draft |
| Ming Ho | 4/5/2016 | 0.02 | Updates during the call with EM team:  Add enhancement to query getCktAssetDetailByCktId |
| Ming Ho | 4/11/2016 | 0.03 | Updates during AID/HLD review: Corrected field name from ucpeCircuitType to uspeServiceLine |
| Ming Ho | 5/19/2016 | 0.04 | 289116-upd01:  Update HLD for getAssetDetails to address test issue of “No managedBy returned when isVms=Y” |
| Ming Ho | 5/24/2016 | 0.05 | Per request from GCp team, change “nonNormalized CktID” 'UnNormalized ATT Circuit ID' where it occurred. |
|  |  |  |  |
| Ming Ho | 6/17/2016 | 0.01 | <270843d> Initial draft |
| Ming Ho | 6/30/2016 | 0.02 | <270843d-upd1>  Input form INSTAR: need to use Logical PAID to trigger INSTAR WS, in order to get complete data set. Updated HLD accordingly. |
| Ming Ho | 8/23/2016 | 0.03 | <270843d-upd2>   * As of 8/23/2016, INSTAR AID has Layer3Info with occurrence 1-1, but WSDL has 1-N. * This field will stay unchanged (matching Instar AID); * GCP code will need to pick up the first Layer3Info instance and get the InterfaceString value. |
|  |  |  |  |
| Ming Ho | 9/29/2016 | 0.01 | <289207>  Enhance web service getIPSvcAccessPointDetailByWanLinkIpAddr:  Add GigEthernetIndicator  Re-arrange the new access circuit structure. |
| Ming Ho | 10/5/2016 | 0.02 | <289207> Minor updates during HLD review. Complete. |
| ~~Ming Ho~~ | ~~12/2/2016~~ | ~~0.03~~ | ~~<289207-QC85019> Address issue with blank customer IP address field for the physical port associated with an ‘Nx100GigE’ or ‘Nx10GigE’ bundle.~~ |
| Ming Ho | 4/13/2017 | 0.03 | <289207-QC176237> Address issue with blank ioc1 field for the <NxXGigEBundleList> structure, update Step 300 |
| Ming Ho | 4/28/2017 | 0.04 | <289207-CR164691>  Add fields to NxXGigEBundleList structure in query getIPSvcAccessPointDetailByWanLinkIpAddr  (API: InquireTroubledNetworkAssetSummary) |
| Ming Ho | 5/8/2017 | 0.05 | <289503 CR165525> was created to replace CR164691 |
| Archana Gattu | 12/7/2017 | 0.06 | Updates for PID 297100a for transactionType: "IPSAPDetailByWanLinkIPAddress(legacy name getIPSvcAccessPOintDetailByWanLinkIpAddr) |
|  |  |  |  |
| Ming Ho | 3/1/2018 | 0.01 | <301033> US374866  Re-engineer EDF logic to use the A&AI data sourced from DMaaP instead of the A&AI batch feed.  Support Vyatta uCPE. |

Table of Contents

[Overview 3](#_Toc468455410)

[Problem Statement 3](#_Toc468455411)

[Project Description: 3](#_Toc468455412)

[Design Decisions 3](#_Toc468455413)

[Change Summary <289207>: 3](#_Toc468455414)

[Change Summary <270843d>: 4](#_Toc468455415)

[getIPSvcAccessPointDetailByWanLinkIpAddr – 289207 5](#_Toc468455416)

[Common logic – convert circuit format: CircuitFormatConversion\_ICORE\_to\_CLCI\_forNise – (created for 286284, with small change) 26](#_Toc468455417)

[Alternative Designs 28](#_Toc468455418)

[Assumptions/Risks 28](#_Toc468455419)

[Traceability Matrix 28](#_Toc468455420)

[Pre-Production Disaster Recovery Planning 28](#_Toc468455421)

[Other Plans and References 28](#_Toc468455422)

[Acceptance & Approvals 28](#_Toc468455423)

## Overview

The High Level Design (HLD) describes how an application will implement the architectural concept and meet the requirements allocated to that application. The High Level Design describes the functionality the application will provide, the overall design for providing that functionality and meeting the nonfunctional requirements, and the rationale for choosing that design. The High Level Design also describes how the application will cooperate and interface with other applications to provide an integrated solution that achieves the architectural concept.

The High Level Design covers these topics:

* Problem Statement
* Design Decisions
* Alternative Designs
* Assumptions/Risks
* Traceability Matrix
* Other Plans/References

## Problem Statement

### Project Description:

See SR document.

## Design Decisions

### Change summary <297100a>:

|  |  |
| --- | --- |
| **Query Name** | **Changes** |
| getIPSvcAccessPOintDetailByWanLinkIpAddr | <USnumber>  EDF to enhance its existing service to provide the below 2 new fields   * vpePtniiName is populated only when the input CR IP associated to vPE or other DriveNet/Arista device. * AOTS TM users can use this field as indicator whether Drivenet/Arista device is involved, and use value in this field as identifier to trigger the INOD APIs.   misOnVpeFlag  vpePtniiName |

### Change Summary <289207>:

|  |  |
| --- | --- |
| **Query Name** | **Changes** |
| getIPSvcAccessPOintDetailByWanLinkIpAddr | <AID-289207-GCP/AOTS-TM 010> map to GCP User Story US765358  US765358 289207\_GCP\_EDF PROCESS Nx100G Requests AOTS\_TM  Parent F27221, F27224  GCP will process requests for Nx100G services, so that the AOTS\_CM can recognize the Circuits that comprise the Nx100G Bundle.  The affected api getIPSvcAccessPointDetailByWanLinkIpAddr  Assumption: Input from AOTS\_TM ip address of the Nx10Gige or Nx100Gige service  Assumption: LEC\_CKT\_ID and Tie Down Information is in NISE\_CORE  GCP/EDF shall add 2 new data items to the output   * Nx100GigEthernet Indicator * Nx10GigeEthernet Indicator   GCP/EDF shall recognize the input Ip address from AOTS\_TM is for an Nx100GigEthernet or Nx10GigEthernet Service  Note: Nx10GigEthernet indicator = ‘Y’ when ip\_serv\_acc\_pt.LAG\_TYPE = ‘Nx10GigE’  Nx100GigEthernet indicator = ‘Y’ when ip\_serv\_acc\_pt.LAG\_TYPE = ‘Nx100GigE’  Note: The Security Details Data that will be received from the InquireDeviceSecurityDetails shall apply only to the Bundle Logical Paid for the Nx100G, Nx10G service.  Note:  GCP/EDF shall provide the list of all CLCIs that make up the Nx10G or Nx100G service and shall provide the related data for:  ip\_serv\_acc\_pt.LAG\_SDID for the Bundle will match all associated physical links which will allow the collection of the physical links.  instar.cust\_access.ioc1 will be the att circuit that will be used to query NISE\_C to get the LEC\_CKT\_ID and Tie\_Down Information   * LEC\_CKT\_ID * Tie\_Down Information   Success Criteria  GCP/EDF receives a request for an Nx100G or Nx10G service using the Web service listed above**,** GCP/EDF returns:   * Nx10GigEthernet Indicator * Nx100GEthernet Indicator * The list of CLCIs for the Nx10Gig or Nx100G service * The Security Details that is associated with the logical PAID of the Bundle. * The Tie\_Down Information for each CLCI in the Bundle. * The lEC\_CKT\_Id for each CLCI in the Bundle. |

### Change Summary <270843d>:

|  |  |
| --- | --- |
| **Query Name** | **Changes** |
| getIPSvcAccessPOintDetailByWanLinkIpAddr | <HLD-270843d-GCP/AOTS-TM 010> map to GCP User Story US649783  **Project 270843d**      also F20426, F20434  GCP shall add attributes to **getIPSvcAccessPointDetailByWanLinkIpAddr**  **The new data elements include:**  **From INSTAR**   * PE L3 IP address  (source will be INSTAR MV\*) * Physical Interface String   (Interface<TenGigE> $rack/$slot/0/$port/$Y) [Min 1, Max 1] (Source will be INSTAR MV \*) * MTU IPv4 * MTU Ipv6   **From NISE-CORE - using the CLCI of the ACCESS\_CKT:**  Layer 1 Tie Down   (source NISE\_C) (will be Working Tie Down and Protection Tie Down)   * + Location CLLI     - TERM  (CHF\_CIRCUIT\_TERM\_A) or (CHF\_CIRCUIT\_TERM\_Z) whichever one has the Tie Down   + Working  TD FIC Code with panel and port     - WKG\_TD  (CHF\_TDI\_WORKING\_A) or (CHF\_TDI\_WORKING\_Z)   + Protection TD  FIC Code with panel and port     - PROT\_TD  (CHF\_TDI\_PROTECTION\_A) or (CHF\_TDI\_PROTECTION\_Z)   + LEC\_CKT\_ID   + ACCESS\_VENDOR   Type of Circuit – OTN, ETH, TDM  From INSTAR WS - **getLayer3Data using PAID**  Vlan Logical Interface String (Interface<TenGigE> ($rack/$slot/0/$port/$Y.$VLAN\_ID, ex: InterfaceTenGige0/2/0/3/9.200 ) – [Min 0, Man ?]   * + - MultiBGP Indicator (Derivable – If more than one BGP Session, then Yes)     - MultiWan Indicator (Derivable – If more than one WAN, then Yes)     - Packet Filter Type     - COS Data       * Ingress ProfileId ( example of default values 40/100G   ‘880’, 1/10G  ‘190’  and others)       * Ingress Profile Name       * Egress ProfileId       * Egress Profile Name       * Default Tunnel Class Indicator (Y,N)     - BGP Data       * BGP Session: (Repeat) (Min 1, Max ?)         + BGP Type         + Routing Protocol         + Neighbor IP - IPv4, IPv6         + BFD  Indicator (For Each BGP Session) (derivable if BfdHeartbeat   found then Yes)         + BFD Interval (for each BGP Session)  <ns4:BfdHeartBeat>1000</ns4:BfdHeartBeat> example         + Route Filter Type (1:1)  for each BGP Session,  will have one of the following values: TBTF, PLR, CUST-FACE         + Next-Hop Address (0:1) for each BGP Session “Loopback Address” |

### getIPSvcAccessPointDetailByWanLinkIpAddr – 289207

|  |  |
| --- | --- |
| **Query Name** | getIPSvcAccessPointDetailByWanLinkIpAddr  API name: **InquireTroubledNetworkAssetSummary** |
| **PIDs, CRs** | IPSvcAccessPointDetail - PA769(R12) (P2A94) (CR PC880.36) (PE755) (PF542) (CR PF542.14), PF783, 121050, 118006c, 176282, 178941, 220569, 188263, 231615, 232213, 261031a, 287954a, 289116.141314, 270843d, 289207, 289207-QC176237, ~~289207-CR164691~~, 289503 CR165525 |
| **Service Type** | Managed |
| **Data Source** | INSTAR, ICORE, CLARIFY, NSI, SiDBOR, A&AI, NOSE-Core |
| **WSDL File** | DBORIPSvcContract.wsdl |
| **Client App** | AOTS TM, EM |

**Query Change Summary:**

|  |  |
| --- | --- |
| **Project ID / Ticket** | **Change Summary / Notes** |
| PA769  <AOTS Rel.6> | Add new fields to output: Connection\_Transport\_Network, Migration\_Indicator, Hybrid |
| 176181-Pivot R1 | **Note:**  **<176181-GCP-AOTS Req.#110>**  For the Pivot R1 project, the following changes are made in this method:   * Added PivotIndicator to the output structure. This field is derived from INSTAR. ip\_serv\_acc\_pt.ati\_value * When retrieving CUST\_ACCESS.acc\_ckt, if it’s blank, use CUST\_ACCESS.ioc1 to populate the custAccessCktId field. |
| 231615 | **Note on access logic change:**   * Retrieve IP\_SERV\_ACC\_PT. SERV\_ACC\_PT\_ID field for the input IP Address – add this attribute in existing logic * Use the IP\_SERV\_ACC\_PT. SERV\_ACC\_PT\_ID to determine the CustomerAssetAliasName |
| Ticket-149268858, GCP IE 149756496 | **Issue:** For the following fields are not populate even though data is available in the source - ICORE  accessTailTech  accessInterTech  Test Data:   |  |  |  | | --- | --- | --- | | CA.ACC\_CKT | ACCESSTAILTECH | ACCESSINTERTECH | | MLEC.981411..ATI | SEAMLESS CONTROL PLANE | MPLS INTER-AS | |
| 234531b uat  June-2013 | Added note to fields specified by AOTS |
| 261031a  Feb'2014 | Updated INSTAR logic to add condition for checking IPv6 full format |
| 287954a | <HLD-287954a-GCP/AOTS-TM 010> map to GCP User Story US553577   * Add Cascaded uCPE Indicator at access circuit level * Allower AOTS to call INOD DeviceDetails API for Cascaded uCPE Indicator=Y. |
| 289116.141314  July’2016 | <HLD-289116.141314-GCP-EM 020> Map to US640793- CR141314 - US GCP-EM supporting blocking transport ticketing  In the GCP-EM interface, we need to enhance the following existing WebServices:  For MIS : getIPSvcAccessPointDetailByWanLinkIpAddr  (input : wanLinkIpAddr), Add   * uCPE indicator – existing as isCascadedUCpe * uCPE hostname – existing as ucpeIdentifier * managed by – added ucpemanagedBy |
| 270843d  Oct’2016 | <HLD-270843d-GCP/AOTS-TM 010> map to GCP User Story US649783 (F20426, F20434)  For getIPSvcAccessPointDetailByWanLinkIpAddr WebService for AOTS TM, the additional data that will be in support of Security Features for project 270843d.  **The new data elements include:**  **From INSTAR**   * PE L3 IP address  (source will be INSTAR MV\*) * Physical Interface String   (Interface<TenGigE> $rack/$slot/0/$port/$Y) [Min 1, Max 1] (Source will be INSTAR MV \*) * MTU IPv4 * MTU Ipv6   **From NISE-CORE - using the CLCI of the ACCESS\_CKT:**  Layer 1 Tie Down   (source NISE\_C) (will be Working Tie Down and Protection Tie Down)   * + Location CLLI     - TERM  (CHF\_CIRCUIT\_TERM\_A) or (CHF\_CIRCUIT\_TERM\_Z) whichever one has the Tie Down   + Working  TD FIC Code with panel and port     - WKG\_TD  (CHF\_TDI\_WORKING\_A) or (CHF\_TDI\_WORKING\_Z)   + Protection TD  FIC Code with panel and port     - PROT\_TD  (CHF\_TDI\_PROTECTION\_A) or (CHF\_TDI\_PROTECTION\_Z)   + LEC\_CKT\_ID   + ACCESS\_VENDOR   Type of Circuit – OTN, ETH, TDM  From INSTAR WS - **getLayer3Data using PAID**  Note: The api will return data for only one interface string  If the input is for a Physical PAID, the api may return limited data  Input Physical PAID is ARPA then BGP Session Data will be returned  Input Physical PAID has a VLAN, then Session Data will not be returned  Input LogicaL PAID, then Session Data will be returned along with Logical Interface String  Vlan Logical Interface String (Interface<TenGigE> ($rack/$slot/0/$port/$Y.$VLAN\_ID, ex: InterfaceTenGige0/2/0/3/9.200 ) – [Min 0, Man ?]   * + - MultiBGP Indicator (Derivable – If more than one BGP Session, then Yes)     - MultiWan Indicator (Derivable – If more than one WAN, then Yes)     - Packet Filter Type     - COS Data       * Ingress ProfileId ( example of default values 40/100G   ‘880’, 1/10G  ‘190’  and others)       * Ingress Profile Name       * Egress ProfileId       * Egress Profile Name       * Default Tunnel Class Indicator (Y,N)     - BGP Data       * BGP Session: (Repeat) (Min 1, Max ?)         + BGP Type         + Routing Protocol         + Neighbor IP - IPv4, IPv6         + BFD  Indicator (For Each BGP Session) (derivable if BfdHeartbeat   found then Yes)         + BFD Interval (for each BGP Session)  <ns4:BfdHeartBeat>1000</ns4:BfdHeartBeat> example         + Route Filter Type (1:1)  for each BGP Session,  will have one of the following values: TBTF, PLR, CUST-FACE         + Next-Hop Address (0:1) for each BGP Session “Loopback Address”   **Use Logical PAID to trigger this INSTAR WS.**  **Peter 6/7/2016:**  The new api will only return      Route Filter Type      Packet Filter Type  It will not have packet filter name or route filter name.  **Note:** The WAN,BFD Interval,BGP,Route Filters,Default VLAN Tunneling is considered logical/config data and shall be retrieved using a new Federated API which will access  INSTAR CSI api (getLayer3Data |
| 289207  Feb’2017 | <AID-289207-GCP/AOTS-TM 010> map to GCP User Story US765358  US765358 289207\_GCP\_EDF PROCESS Nx100G Requests AOTS\_TM  GCP will process requests for Nx100G services, so that the AOTS\_CM can recognize the Circuits that comprise the Nx100G Bundle.  GCP/EDF shall recognize the input Ip address from AOTS\_TM is for an Nx100GigEthernet or Nx10GigEthernet Service  Add 2 new data items to the output, for ip\_serv\_acc\_pt.LAG\_TYPE = ‘Nx10GigE’, ‘Nx100GigE’   * Nx100GigEthernet Indicator * Nx10GigeEthernet Indicator   Assumption: Input from AOTS\_TM ip address of the Nx10Gige or Nx100Gige service  Assumption: LEC\_CKT\_ID and Tie Down Information is in NISE\_CORE  Note:  GCP/EDF shall provide the list of all CLCIs that make up the Nx10G or Nx100G service and shall provide the related data for:  ip\_serv\_acc\_pt.LAG\_SDID for the Bundle will match all associated physical links which will allow the collection of the physical links.  instar.cust\_access.ioc1 will be the att circuit that will be used to query NISE\_C to get the LEC\_CKT\_ID and Tie\_Down Information   * LEC\_CKT\_ID * Tie\_Down Information   Success Criteria  GCP/EDF receives a request for an Nx100G or Nx10G service using the Web service listed above**,** to return:   * Nx10GigEthernet Indicator – use one field to display the two values instead; * Nx100GEthernet Indicator * The list of CLCIs for the Nx10Gig or Nx100G service * The Tie\_Down Information for each CLCI in the Bundle. * The lEC\_CKT\_Id for each CLCI in the Bundle. * The Security Details (from INSTAR web service) that is associated with the logical PAID of the Bundle, apply only to the Bundle Logical Paid for the Nx100G, Nx10G service. – provided for 270843d, 1610 project.   <289207-QC85019> Address issue with blank IP address field for the physical port associated with an ‘Nx100GigE’ or ‘Nx10GigE’ bundle.  4/13/2017:  <289207-QC176237> Address issue with blank ioc1 field for the <NxXGigEBundleList> structure, update Step 300 |
| 289503 CR165525  ~~289207-CR164691~~  July’2017 | <HLD-289503 CR165525-GCP/AOTS-TM 010> map to GCP User Story US275090  ~~<HLD-CR164691-GCP/AOTS-TM 010> map to GCP User Story US275090~~  To address NX100GE PVT issue - problem with the display of order information on AOTS ticket,  Add to the NxXGigEBundleList   * Router\_Information (complex)   + IOC1CktId   + EquipmentPTNII   + EquipmentSlot   + EquipmentPort   + EquipmentBasePort |
| 297100a (1804 Rel) | Add misOnVpeFlag and vpePtniiName new fields on output structure.   * vpePtniiName is populated only when the input CR IP associated to vPE or other DriveNet/Arista device. * AOTS TM users can use this field as indicator whether Drivenet/Arista device is involved, and use value in this field as identifier to trigger the INOD APIs. |
| <301033>  June’2018 | US374866  Re-engineer EDF logic to use the A&AI data sourced from DMaaP instead of the A&AI batch feed.  Support Vyatta uCPE. |

**Input: CustRouterIPAddress**

|  |  |  |
| --- | --- | --- |
| **Input Attribute** | | **NOTES** |
| CommonHeader | String username | User name of the method for authentication/authorization |
| String token | Password of the user |
| CustRouterIPAddress - 2 | Varchar2 (55)  M | CRIPv4Address =CustRouterIPAddress - 2  CRIPv4Address/CRIPv6Address  Query: Logic Ref. Only.  Select INSTR('IP ADDRESS',':') FROM input;  INSTR('IPADDRESS',':') |
| contractType | Varchar2(25)  O | Contract Type  ~~Mandatory for LACNTY~~ |

Output:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Table.Column** | **Source on** GCP/**DBOR side** | **Data Type/Length** |
| CustRouterIPAddress | input from AOTS | input from AOTS |  |
| ipCustId | Premise.CustId | Premise.CustId | varchar2(20) |
| MCN | Retrieved from Clarify |  | varchar2(10) |
| svcAccess | IP\_SERV\_ACC\_TYPE.acc\_type | Sap.ServiceAccess | varchar2(20) |
| pivotIndicator | Derived from ip\_serv\_acc\_pt.ati\_value | <176181> Valid values: Y, N | varchar2(1) |
| custAccessCktId | CUST\_ACCESS.acc\_ckt | Sap.CustAccess.CustAccessCkt  <287954a> When isCascadedUCpe=Y, the value in this field can be used to call the API retrieving uCPE info | varchar2(25) |
| ioc1CktId | CUST\_ACCESS.ioc1 | Sap.CustAccess.IOC1 | varchar2(25) |
| accessArrangement | ACCESS\_ARGMT.access\_argmt | Sap.AccessArrangement | varchar2(30) |
| transportNetwork | TRANSPORT\_NETWORK.name | Sap.TransportNetwork | varchar2(15) |
| isCascadedUCpe | Derived | <287954a> Valid values  Y – The input router (represented by the WAN IP) cascades to a uCPE device  N - default | varchar2(1)  R |
| ucpeIdentifier | ~~A&AI.Device.ptnii\_Name~~  A&AI.aai\_pserver.hostname <301033> | <287954a>   * This is the equipment PTNii returned from A&AI data, which will be the identifier (ptnii or host name) of the uCPE. * When isCascadedUCpe=Y, this field is populated. Otherwise it’s null. * This is not for AOTS to display, but to use it to call the INOD DeviceDetails API, to retrieve uCPE info. | varchar2(50)  O |
| ucpeManagedBy | ucpeManagedBy  A&AI.aai\_pserver.management\_option <301033> | <289116.141314-US640793> Support EM:  To show the UCPE is managed by ATT or customer | varchar2(50)  O |
| misOnVpeFlag | Derived value using Instar data | <297100a 1804Rel> | Boolean  Valid Values:  True  False  O |
| vpePtniiName | Instar.equip.ptnii\_equip\_name | <297100a 1804 Rel>  VPE Identifier | Varchar2(20)  O |
| peRouterIpAddress | Instar.serial\_ip\_addr.ip\_address  or  Instar.ipv6\_addigned\_link\_ips.ipv6\_address | <270843d> | Varchar2(45)  O |
| interfaceString | Instar.ip\_port\_adgmt.interface\_string | <270843d> This is the port physical interface string | Varchar2(35)  O |
| mtuIpv4 | Instar.IP\_serv\_acc\_pt.mtu\_value | <270843d>  Maximum Transfer Unit Speed | Number  O |
| mtuIpv6 | Instar.IP\_serv\_acc\_pt.v6\_mtu\_value | <270843d>  Maximum Transfer Unit Speed | Number  O |
| gigEthernetLagType | Instar.Ip\_serv\_acc\_pt.lag\_type | <289207>  To recognize the input Ip address is for an Nx100GigEthernet or Nx10GigEthernet Service, with values:  Nx10GigE  Nx100GigE  Null – for other services | Varchar2(10)  O |
| <NxXGigEBundleList> | Occurrence: 0:12 | <289207> This circuit list is only populated when the GigE Lag Type has values of Nx10GigE or Nx100GigE | Complex |
| **<Layer1TieDown>** | Occurrence: 0:1 | <270843d> | Complex |
| **<VlanLayer3Data>** | Occurrence: 0:1 | <270843d> | Complex |
| IPPremiseDetail |  |  |  |
| custName | Premise.CustName | Premise.CustName | varchar2(50) |
| address | Premise.Address | Premise.Address | varchar2(75 |
| Address2 | Premise. Address2? | Premise. Address2? | varchar2(75) |
| city | Premise.City | Premise.City | varchar2(26) |
| state | PREMISE.state\_abbr | Premise.State | varchar2(2) |
| zip | Premise.Zip | Premise.Zip | varchar2(16) |
| countryCode | COUNTRY.country\_abbr | Premise.CountryCode | varchar2(2) |
|  |  |  |  |
| arrayOfIPSvcOption | array | array |  |
| optionName | IP\_SERVICE\_OPTIONS.option\_name | Sap.ServiceOptionList.ServiceOption.OptionName | varchar2(40) |
| svcStatus | Sap.ServiceOptionList.ServiceOption.  SvcStatus | Sap.ServiceOptionList.ServiceOption.SvcStatus | varchar2(30) |
|  |  |  |  |
| arrayOfIPSvcAttribute |  | Sap.ServiceOptionList.ServiceOption.OptionAttribute |  |
| attributeName | SERVICE\_OPTION\_VALUES.name | Sap.ServiceOptionList.ServiceOption.OptionAttribute.AttributeName |  |
| attributeValue | SERVICE\_OPTION\_VALUES.value | Sap.ServiceOptionList.ServiceOption.OptionAttribute.AttributeValue | varchar2(50) |
| trafficProfileName | TRAFFIC\_PROFILE.traffic\_profile\_name | Sap.ServiceOptionList.ServiceOption.OptionAttribute.TrafficProfileName | varchar2(50) |
|  |  |  |  |
| ConnectionTransport\_network (PA769) | IP\_PORT\_ASGMT.Conn\_Trans\_Net | IP\_PORT\_ASGMT.Conn\_Trans\_Net  Possible values:  ASI or cSBC or cAT&T or AT&T or  Alascom or AGN or GFN | varchar2(30) |
| MigrationIndicator (PA769) | IP\_SERV\_ACC\_PT.migration\_Ind | IP\_SERV\_ACC\_PT.migration\_Ind  Possible values:  cSBC DIA-OOR or cSBC DIA-IR or cSBC DIA-IR ASSIGNED | varchar2(25) |
| EBondInd | IPRouterData.ebondind,  IPCustomer.ebondind,  IPMCN.ebondind |  | Varchar2(1)  Valid values = Y , N |
| Hybrid (PA769) | EQUIPMENT.Hybrid | EQUIPMENT.Hybrid | varchar2(40) |
| <P2A94> |  |  |  |
| contractType | DBOR NSI:Valid Values = NETWORX | DBOR NSI:Valid Values = NETWORX  Populate with ‘NETWORX’ if the networx specific attributes are found in the NETWORX\_MCN\_LIST table. | Varchar2(25) |
| (231615) customerAssetAliasName | alias\_value.value | GCP gets data from EM or BDMAP, and/or will store in a schemas called VIRTEXT (virtual extension) | Varchar2(50)  optional |
| assetObjectId(Long) | GDB asset table  asset.id | gdbAssetObjectID  2/17/2017 – Included this field from the AID. | Number(20)  optional |
|  |  |  |  |
| **NETWORXInfo** |  |  |  |
| accessInterTech | ICORE.Intercon\_tech.intercon\_techname | <Ticket-149268858> | varchar2(25) |
| accessTailTech | ICORE.Access\_method.access\_method\_desc | <Ticket-149268858> | varchar2(25) |
| agencyName | NETWORX\_MCN\_LIST.Customer\_Name |  | Varchar2(40) |
| agencyID | NETWORX \_MCN\_LIST.Agency\_Hierarchy\_Code |  | Varchar2 (8) |
| MCN | FACILITY\_ACCOUNT.mcn |  | Varchar2(9) |
| GRC | FACILITY\_ACCOUNT.grc |  | Varchar2(3) |
| SOC | FACLITY\_ACCOUNT.soc |  | Varchar2(3) |
| TSP |  |  |  |
| CSI |  |  |  |
| Ubi |  |  |  |

**<**NxXGigEBundleList**>**

Occurrence: 0:N (N can be up to 12)

<289207> This circuit list is only populated when the GigE Lag Type has values of Nx10GigE or Nx100GigE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Attribute Name** | **R/O/C** | **Data Type and Length** | AOTS Field Mapping  **GUI Label** | **Data Source** | **Comments** |
| ioc1CktId | O | Varchar2(64) | Circuit ID (IOC) | instar.cust\_access.ioc1 |  |
| equipmentPtnii | O | Varchar2(20) | Equipment PTNII | instar.equipment.ptnii\_equip\_name | ~~<289207-CR164691>~~  <289503 CR165525> |
| equipmentSlot | O | Number(10) | Slot | instar.component.Beg\_slot | ~~<289207-CR164691>~~  <289503 CR165525> |
| equipmentPort | O | Number(10) | Port | instar.component.Port | ~~<289207-CR164691>~~  <289503 CR165525> |
| equipmentBasePort | O | Number(10) | Base Port | instar.component.Base\_port | ~~<289207-CR164691>~~  <289503 CR165525> |
| <Layer1TieDown> | O | Complex | Occurrence: 0:N |  |  |

**<Layer1TieDown>** <270843d>

Occurrence: 0:1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **R/O/C** | **Data Type and Length** | **Data Source** | **Comments** |
| circuitId | O | Varchar2(25) | Nice-core.circuit\_has\_fact.  chf\_app\_circuit\_nm | The access circuit in CLCI format |
| circuitType | O | Varchar2(10) | Derived from Circuit SVC code |  |
| locationClli | O | Varchar2(11) | Nice-core.Circuit\_has\_fact.  Chf\_circuit\_term\_a or Chf\_circuit\_term\_z | whichever one (A or Z end) has the Working Tie Down |
| workingTieDown | O | Varchar2(20) | Nice-core.circuit\_has\_fact.  chf\_tdi\_working\_a, or chf\_tdi\_working\_z | FIC Code with panel and port. |
| protectionTieDown | O | Varchar2(20) | Nice-core.circuit\_has\_fact.  chf\_tdi\_protection, or chf\_tdi\_protection\_z | FIC Code with panel and port. |
| lecCircuitId | O | Varchar2(25) | Nice-core. Derived |  |
| accessVendor | O | Varchar2(30) | Nice-core.circuit\_has\_fact.  Chf\_Access\_vendor\_a or  Chf\_Access\_vendor\_z |  |

**<VlanLayer3Data>** <270843d> Data from INSTAR WS/PI, and some fields are derived.

Complex

Occurrence: 0:1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **R/O/C** | **Data Type and Length** | **Data Source** | **Comments** |
| isMultiBgp | O | Varchar2(1) | Derived. | Valid values:  Y – more than one BGP sessions from Instar API call;  N – zero or one BGP session returned. |
| isMultiWan | O | Varchar2(1) | Derived. | Valid values:  Y – more than one "ns4:WanIPAddress" occurrences in InterfaceAddress sessions from Instar API call;  N – zero or one "ns4:WanIPAddress" occurrences in InterfaceAddress sessions |
| vlanLogicalInterfaceString | O | Varchar2(255) | Instar.getLayer3Data.Layer3Info | In Layer3Info.InterfaceString  **Note:** <270843d-upd2>   * As of 8/23/2016, INSTAR AID has Layer3Info with occurrence 1-1, but WSDL has 1-N. * This field will stay unchanged (matching Instar AID); * GCP code will need to pick up the first Layer3Info instance and get the InterfaceString value. |
| **<ClassOfService>** | O | Complex  Occurrence: 0:1 |  |  |
| **<BgpProtocolData>** | O | Complex  Occurrence: 0:n |  |  |

**<ClassOfService> <270843d>**

Complex

Occurrence: 0:1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **R/O/C** | **Data Type**  **and Length** | **Data Source** | **Comments** |
| ingressProfileId | O | Number | Instar.getLayer3Data | From ClassOfService.IngressProfileId |
| egressProfileId | O | number | Instar.getLayer3Data | From ClassOfService.EgressProfileId |
| cosPackage | O | number | Instar.getLayer3Data | From ClassOfService.CosPackage |
| cosPolicyType | O | Varchar2(25) | Instar.getLayer3Data | From ClassOfService.CosPolicyType |
| defaultTunnelClassIndicator | O | Varchar2(1) | Instar.getLayer3Data | From ClassOfService.DefaultTunnelClassIndicator |

**<BgpProtocolData> <270843d>**

Complex

Occurrence: 0:n

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **R/O/C** | **Data Type**  **and Length** | **Data Source** | **Comments** |
| bgpType | O | Varchar2(15) | Instar.getLayer3Data | From BgpProtocolData.BgpType |
| routingProtocol | O | Varchar2(15) | Instar.getLayer3Data | From BgpProtocolData.RoutingProtocol |
| bfdIndicator | O | Varchar2(1) | Derived | From BgpProtocolData.MisBgpParameter.BfdHeartBeat  Values:  Y - If this value populated  N – otherwise |
| packetFilterType | O | Varchar2(10) | Instar.getLayer3Data | From BgpProtocolData.AclCategory.  Sample values: SAA, BFC |
| **<MisBgpParameter>** | O | Complex  Occurrence: 0:n |  |  |

**<MisBgpParameter> <270843d>**

Complex

Occurrence: 0:n

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Attribute Name** | **R/O/C** | **Data Type**  **and Length** | **Data Source** | **Comments** |
| routerFilter | R | Varchar2(10) | Instar.getLayer3Data | From BgpProtocolData.MISBgpParmeter.RouterFilter |
| bfdInterval | O | Number | Instar.getLayer3Data | From BgpProtocolData.MISBgpParmeter.BfdHeartBeat |
| neighborAddress | O | Varchar2(45) | Instar.getLayer3Data | From BgpProtocolData.MISBgpParmeter.NeighborAddress.ipAddress |

Data Access Logic:

**INSTAR GetDetails XML Query**

Document P7A69 SRD

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Description** | **Required** |
| PortAssignmentId | INSTAR Port Assignment Id | Required if WanLinkIpAddress and GroupId are not specified |
| AssignStatus | Assign Status for a port assignment. E.g. PRINCIPAL | Optional when PortAssignmentId is specified |
| WanLinkIpAddress | Physical Wan Link IP Address of port assignment. | Required if PortAssignmentId and GroupId are not specified |
| GroupId | Group ID. Numeric. | Required if PortAssignmentId and GroupId are not specified |
| GroupStatus | Group Status for a group. e.g. PENDING ADD | Optional when GroupId is specified |

* AOTS will send a Customer Router IP Address and INSTAR uses WAN Link IP Address, so GCP/DBOR will subtract 2 from the last octect of the input from AOTS by 2, for IPv4.
* e.g. 12.118.126.80 - Customer Router IP, 12.118.126.78 is its Wan Link IPaddress

\*\*\* As of November 2006 **GetDetails** has been converted to get data from GCP/DBOR local database.

**Data Access Logic**

1. Retrieve the data from replicated INSTAR tables available in GCP/DBOR using the following logic.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| bids\_dba.SERIAL\_IP\_ADDR,  ipd\_dba.IP\_PORT\_ASGMT,  ipd\_dba.IP\_SERV\_ACC\_PT,  bids\_dba.SITE,  ipd\_dba.CUST\_ACCESS,  bids\_dba.CUSTOMER,  bids\_dba.PREMISE,  bids\_dba.COUNTRY,  ipd\_dba.IP\_SERV\_ACC\_TYPE,  ref\_dba.TRANSPORT\_NETWORK,  ipd\_dba.ACCESS\_ARGMT  <270843d>  bids\_dba.SERIAL\_IP\_ADDR serial\_ip\_addr2, | PREMISE.cust\_id,  PREMISE.address, PREMISE.city, PREMISE.state\_abbr, PREMISE.zip, COUNTRY.country\_abbr,  CUSTOMER.cust\_name,  NVL(CUST\_ACCESS.acc\_ckt, CUST\_ACCESS.ioc1) acc\_ckt,  CUST\_ACCESS.ioc1), IP\_PORT\_ASGMT.assgn\_status,  IP\_SERV\_ACC\_TYPE.acc\_type, ACCESS\_ARGMT.access\_argmt, TRANSPORT\_NETWORK.name,  IP\_SERV\_ACC\_PT. serv\_acc\_pt\_id,  IP\_PORT\_ASGMT.sdid  SERIAL\_IP\_ADDR.ip\_address  <PF220>  IP\_SERV\_ACC\_PT.ati\_value  IP\_SERV\_ACC\_PT. SERV\_ACC\_PT\_ID <231615>  <270843d>  Serial\_ip\_addr2.ip\_address AS peRouterIpAddress,  IP\_serv\_acc\_pt.mtu\_value AS mtuIPv4,  IP\_serv\_acc\_pt.v6\_mtu\_value AS mtuIPv6,  <270843d-upd1>  Substr (ip\_port\_asgmt.interface\_string, 1,  Decode (instr(ip\_port\_asgmt.interface\_string, '.'),  0, length(ip\_port\_asgmt.interface\_string),  instr(ip\_port\_asgmt.interface\_string, '.') -1))  AS interfaceString,  IP\_port\_asgme.ip\_port\_asgmt\_id AS logicalPAID, used to call INSTAR getLayer3Data WS  <289207>  Ip\_serv\_acc\_pt.lag\_type  Ip\_serv\_acc\_pt.lag\_sdid  </289207> | SERIAL\_IP\_ADDR.ip\_address = input parameter CustRouterIPAddress - 2  AND SERIAL\_IP\_ADDR.serial\_ip\_addr\_id = IP\_PORT\_ASGMT.wan\_addr\_id  AND IP\_PORT\_ASGMT.assgn\_status = 'PRINCIPAL'  AND IP\_PORT\_ASGMT.serv\_acc\_pt\_id = IP\_SERV\_ACC\_PT.serv\_acc\_pt\_id  AND IP\_SERV\_ACC\_PT.site\_id = SITE.site\_id  AND IP\_SERV\_ACC\_PT.cust\_id = CUSTOMER.cust\_id  AND IP\_SERV\_ACC\_PT.site\_id = CUST\_ACCESS.site\_id (+)  AND ~~CUST\_ACCESS.cust\_id = CUSTOMER.cust\_id~~ AND SITE.loc\_id = PREMISE.loc\_id  AND SITE.transport\_network\_id = TRANSPORT\_NETWORK.transport\_network\_id  AND PREMISE.country\_name = COUNTRY.country\_name (+)  AND IP\_SERV\_ACC\_TYPE.serv\_acc\_type\_id = IP\_SERV\_ACC\_PT.serv\_acc\_type\_id  AND IP\_SERV\_ACC\_PT.access\_argmt\_id = ACCESS\_ARGMT.access\_argmt\_id  And  Serial\_ip\_addr2.serial\_ip\_addr\_id (+) = IP\_port\_asgmt.ar\_addr\_id <270843d>  **<176181>**   1. If CUST\_ACCESS.acc\_ckt is blank, populate Acc\_ckt field with CUST\_ACCESS.ioc1 2. If IP\_SERV\_ACC\_PT.ati\_value = ‘PVOT’ then   set pivotIndicator = ‘Y’  Else, set pivotIndicator = ‘N’  Populating pivotIndicator field with the value derived here.  Sample implementation:  **Decode** (IP\_SERV\_ACC\_PT.ati\_value, ‘PVOT’, ‘Y’, ‘N’) PivotIndicator  **</176181>**  <270843d-upd1>  Physical interface string – by removing the ending sequence number, if it exists.  Substr() is used to get the section from beginning of the string, to where ‘.’ Is. Decode is used to check whether ‘.’ Exists.  <289207>  If Ip\_serv\_acc\_pt.lag\_type has values  Nx10GigE  Nx100GigE  Then  Set gigEthernetLagType = lag\_type value  <289207-QC85019> Note: Not for this query.  ~~When IP Address is null for the physical port, Retrieve IP for port is associated with the ‘Nx100GigE’ or ‘Nx10GigE’ bundle~~  ~~If above serial\_ip\_addr.ip\_address is null,~~  ~~Then~~  ~~Select Unique SIA.ip\_address~~  ~~From~~  ~~ip\_serv\_acc\_pt SAP,~~  ~~ip\_port\_asgmt PA,~~  ~~serial\_ip\_addr SIA~~  ~~Where 1=1~~  ~~And SAP.lag\_sdid = <Ip\_serv\_acc\_pt.lag\_sdid>~~  ~~And SAP.sap\_type = ‘Regular’~~  ~~And SAP.serv\_acc\_pt\_id = PA.serv\_acc\_pt\_id~~  ~~And PA.cr\_addr\_id = SIA.serial\_ip\_addr\_id~~  ~~Populate the CustRouterIPAddress field~~  ~~End If~~  Follow STEP-300, with  Ip\_serv\_acc\_pt.lag\_sdid  Ip\_serv\_acc\_pt.lag\_type  Else  Set gigEthernetLagType = Null  ~~Follow BAU logic~~  End If  Follow BAU logic |

1. Query the IP Service Options table with the **serv\_acc\_pt\_id** value retrieved from step 1 to get one or more option name using the below logic.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| ipd\_dba.IP\_SELECTED\_OPTS,  bids\_dba.IP\_SERVICE\_OPTIONS,  ipd\_dba.SELECTED\_OPTS\_REL | Ip\_Service\_Options.option\_name,  Ip\_Selected\_Opts.selected\_opt\_id | SELECTED\_OPTS\_REL.serv\_acc\_pt\_id= serv\_acc\_pt\_id from step 1  AND  IP\_SELECTED\_OPTS. selected\_opt\_id=  SELECTED\_OPTS\_REL. selected\_opt\_id  AND  IP\_SERVICE\_OPTIONS.service\_option\_id = IP\_SELECTED\_OPTS.service\_option\_id |

1. For each **Selected Option ID** retrieved from Step 2, retrieve one or more NAME and VALUE from Service Option Values Table using the below logic.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| bids\_dba.SERVICE\_OPTION\_VALUES,  bids\_dba.TRAFFIC\_PROFILE | Service\_Option\_Values.name, Service\_Option\_Values.value | SERVICE\_OPTION\_VALUES.selected\_opt\_id = Selected Opt Id from Step 2  AND  SERVICE\_OPTION\_VALUES.value = TO\_CHAR(TRAFFIC\_PROFILE.traffic\_profile\_id(+) ) AND  ( SERVICE\_OPTION\_VALUES.name like TRAFFIC\_PROFILE.direction || '%'  or SERVICE\_OPTION\_VALUES.name IS NULL ) |

<P2A94>

Based on the cust\_id From step 1 retrieve the MCN/GRC/SOC from ipd\_dba.facility\_account table from GCP/DBOR replicated copy of INSTAR using the logic below.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| Ipd\_dba.FACILITY\_ACCOUNT | MCN,  GRC,  SOC | FACILITY\_ACCOUNT.cust\_id = input cust\_id and FACILITY\_ACCOUNT.type = ‘Service’ |

1. Based on the MCN/GRC/SOC from step 4 retrieve the agency\_hierarchy\_code and customer\_name from NETWORX\_MCN\_LIST using the logic below.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| NETWORX\_MCN\_LIST | AGENCY\_HIERARCHY\_CODE, CUSTOMER\_NAME,  ‘NETWORX’(Not a table attribute) | NETWORX\_MCN\_LIST.MCN = mcn from step 4  and NETWORX\_MCN\_LIST.GRC = grc from step 4  and NETWORX\_MCN\_LIST.SOC = soc from step 4 |

**<231615> Access logic – Determine CustomerAssetAliasName:**

|  |  |  |
| --- | --- | --- |
| **Table (in SiDBOR)** | **Attribute** | **Condition for selection** |
| **Schema: MetaData**  meta\_column mcol,  meta\_table mtab,  meta\_system msys,  source\_key\_value skeyval,  source\_key skey,  object\_type otyp,  **Schema: Virtext**  alias\_type atyp,  alias\_value aval,  alias\_association aass | aval.value AS customerAssetAliasName | msys.id=mtab.id\_meta\_system  And mtab.id=mcol.id\_meta\_table  And mtab.id=skey.id\_meta\_table  And mcol.id=skeyval.id\_meta\_column  And skey.id=skeyval.id\_source\_key  And msys.name='INSTAR'  And mtab.name='IP\_SERV\_ACC\_PT’  And mcol.name='SERV\_ACC\_PT\_ID’  And skeyval.source\_value=  **<IP\_SERV\_ACC\_PT. SERV\_ACC\_PT\_ID> -- Retrieved from Step 1**  And otyp.name='SOURCE\_KEY'  And Atyp.type='CUSTOMER\_DEFINED\_ ASSET\_TICKET\_ALIAS'  And otyp.id=aass.id\_object\_type  And atyp.id=aass.id\_alias\_type  And skey.id=aass.id\_object  And aval.id=aass.id\_alias\_value |

Note: If any of the Networx specific attributes is not available in GCP/DBOR-NSI table, return the rest of the attributes.

Pass Null for TSP/CSI/UBI.

**script Prototype**

getDetails(PortAssignmnetId, AssignStatus, WanLinkIpAddress ,GroupId, GroupStatus)

<http://roswell.mt.att.com/05g/servlet/main/dispmanpage?MANPAGE=GetDetails>

<PF220>

**Data Access Logic for input/output IPv6 IP Address**

Output Attributes:

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Field Size** | **INSTAR Database mapping** |
| CRIpv6Address | Varchar2(39) | IPV6\_ASSIGNED\_LINK\_IPS  .ipv6\_address\_COMPRESS |
| CustRouterIPAddress |  | SERIAL\_IP\_ADDR.ip\_address  Derived by adding ‘2’ to the last octet of wanLinkIpAddr (Ex: wanLinkIpAddr = ‘20.30.10.248’ custIpAddress = ‘20.30.10.250’ |
| peRouterIpAddress | Varchar2(45) | <270843d>  Instar.serial\_ip\_addr.ip\_address  Or if the above field did not return data,  Instar.ipv6\_addigned\_link\_ips.ipv6\_address |

INPUT : CRIPv4Address

Retrieve attributes from replicated INSTAR tables available in GCP using existing logic.

1. Retrieves IP\_PORT\_ASGMT.sdid. The sdid is used as a key.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| bids\_dba.SERIAL\_IP\_ADDR  ipd\_dba.IP\_PORT\_ASGMT | SERIAL\_IP\_ADDR.ip\_address  IP\_PORT\_ASGMT.sdid |  |

1. Query the Instar IPV6\_PORT\_ASGMT\_MAP and IPV6\_ASSIGNED\_LINK\_IPS table to retrieve the below attributes.

|  |  |  |
| --- | --- | --- |
| **INSTAR Table Name** | **Column Name(s)** | **Condition for selection** |
| IPV6\_PORT\_ASGMT\_MAP,  IPV6\_ASSIGNED\_LINK\_IPS,  <170843d>  IPV6\_PORT\_ASGMT\_MAP AS ipv6\_pa\_map2,  IPV6\_ASSIGNED\_LINK\_IPS AS ipv6\_ips2 | IPV6\_ASSIGNED\_LINK\_IPS  .IPV6\_ADDRESS\_COMPRESS  <270843d>  Ipv6\_ips2.ipv6\_address\_compress AS peRouterIpAddress | IPV6\_PORT\_ASGMT\_MAP.sdid = IP\_PORT\_ASGMT.sdid --From step1  And  IPV6\_PORT\_ASGMT\_MAP. ipv6\_link\_ip\_id = IPV6\_ASSIGNED\_LINK\_IPS.ipv6\_link\_ip\_id  And  IPV6\_PORT\_ASGMT\_MAP. IPV6\_ADDRESS\_TYPE\_ID=’10’  (IPv6\_CR Address)  <270843d>  ipv6\_pa\_map2.sdid = ip\_port\_asgmtI.sdid --From step1  And  ipv6\_pa\_map2.ipv6\_link\_ip\_id = ipv6\_ips2.ipv6\_link\_ip\_id  And  ipv6\_pa\_map2.ipv6\_address\_type\_id = ’9’  (IPv6\_AR Address) |

**INPUT : CRIPv6Address**

1. Query the Instar IPV6\_PORT\_ASGMT\_MAP and IPV6\_ASSIGNED\_LINK\_IPS table to retrieve the below attributes.
2. Retrieves IP\_PORT\_ASGMT.sdid. The sdid is used as a key. See above step

|  |  |  |
| --- | --- | --- |
| **INSTAR Table Name** | **Column Name(s)** | **Condition for selection** |
| IPV6\_ASSIGNED\_LINK\_IPS  IPV6\_PORT\_ASGMT\_MAP  IP\_PORT\_ASGMT | IPV6\_ASSIGNED\_LINK\_IPS . IPV6\_ADDRESS\_COMPRESS  IP\_PORT\_ASGMT.sdid | <261031a> Add condition for checking IPv6 Full format:  (IPV6\_ASSIGNED\_LINK\_IPS.ipv6\_address\_compress = CRipv6\_address input from AOTS  OR  IPV6\_ASSIGNED\_LINK\_IPS.ipv6\_address=CRipv6\_address input from AOTS  )  And  IPV6\_ASSIGNED\_LINK\_IPS.ipv6\_link\_ip\_id=  IPV6\_PORT\_ASGMT\_MAP. ipv6\_link\_ip\_id  And  IPV6\_PORT\_ASGMT\_MAP.sdid = IP\_PORT\_ASGMT.sdid |

Retrieve attributes from replicated INSTAR tables available in GCP using existing logic. And

IP\_PORT\_ASGMT.sdid. The sdid is used as a key

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| bids\_dba.SERIAL\_IP\_ADDR  ipd\_dba.IP\_PORT\_ASGMT | SERIAL\_IP\_ADDR.ip\_address  IP\_PORT\_ASGMT.sdid |  |

</PF220>

**<Ticket-149268858>**

Access ICORE to retrieve accessInterTech and accessTailTech, using the **acc\_ckt** retrieved in step

“1. Retrieve the data from replicated INSTAR tables available in GCP/DBOR” :

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Attribute Name** | **Condition for selection** |
| ICORE.Cust\_access ca,  ICORE.Site s  ICORE.Access\_method am,  ICORE.Port\_asgmt pa,  ICORE.Intercon\_tech it | am.access\_method\_desc as accessTailTech,  it.intercon\_techname as  accessInterTech | Rtrim (ca.acc\_ckt) = **<acc\_ckt>**  AND s.site\_id = ca.site\_id  AND s.intercon\_tech\_id = it.intercon\_tech\_id (+)  AND pa.site\_id = ca.site\_id  AND ca.access\_method\_id = am.access\_method\_id (+)  Populate fields accessInterTech and accessTailTech, respectively |

##### **</Ticket-149268858>**

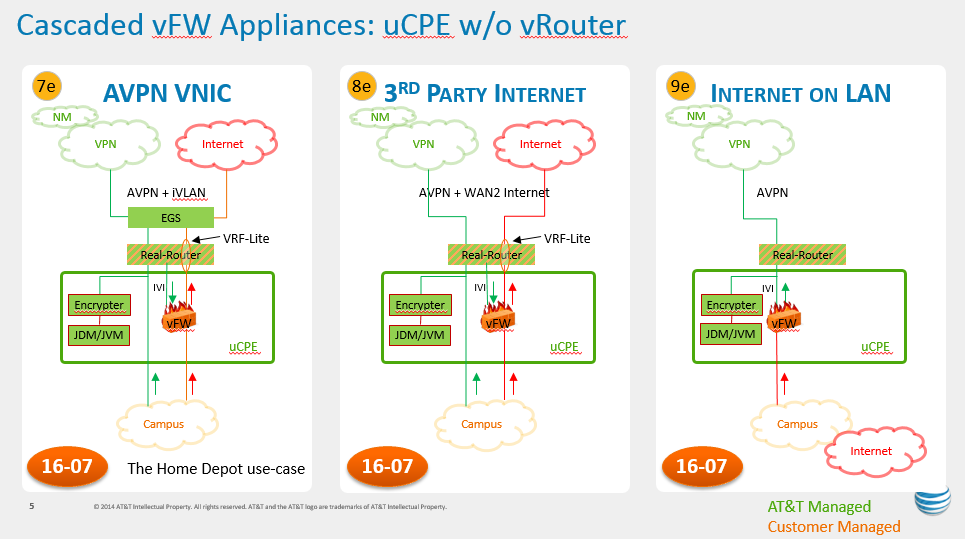
**<287954a> Determine Cascated Indicator (using A&AI schema)**

* isCascatedUCpe field is default to ‘N’
* If isCascatedUCpe field is set to ‘Y’, AOTS can call the INoD DeviceDetails API to retrieve uCPE and vFW info using the ptnii/hostname for uCPE retrieved here.

<301033>

This is to support cascaded uCPE (with vFW, but without vRouter - RT) behind Real Router, LAN side internet, as the diagram shown below. This is uCPE without routing functionality, relying on real router to provide the routing functionality

* This setup has a Real-Router (aka physical , non-virtualized, traditional) in front of the uCPE device. This Router can be AT&T managed or Customer managed.
* If the router is third party, and access is third party then AT&T will not monitor the 3rd party access. Customer will be responsible for 3rd party transport.
* In this configuration, the routing function is done by the real router. The uCPE will have vFW.



|  |  |  |
| --- | --- | --- |
| **Response Fields** | **Source Tables** | **Conditions, Comments** |
|  | Step NOD-100 | **For input Access Circuit ID,**  Input: <custAccessCktId>, retrieved from step 1.  Output: isCascadedUCpe – derived  ucpeIdentifier - this is the hostname of the uCPE  ucpeMamagedBy  Data source: A&AI in EDF  <301033> Note:  New table created in AAI schema containing equipment types for UCPE (including Juniper and Vyatta UCPEs), so no hardcode needed.  ucpe\_equip\_type\_set.ucpe\_equip\_type |
| **Retrieved fields:**  Pserver.hostname: ucpeIdentifier,  Pserver.management\_option: ucpeMamagedBy | STEP NOD-100-A  A&AI Tables:  Aai\_pserver pserver,  Aai\_p\_interface pinterface,  Aai\_p\_interface\_physical\_link pip\_link,  Aai\_physical\_link plink | Derive isCascatedUcpe field – After the access circuit is retrieved.  Input: <custAccessCktId>, retrieved from step 1.  Conditions:  And plink.circuit\_id = <input-assetTag>  And pserver.hostname = pinterface.hostname  And pserver.equip\_type IN  (ucpe\_equip\_type\_set.ucpe\_equip\_type)  And pinterface1.hostname = pip\_link.hostname  And pinterface1.interface\_name = pip\_link.interface\_name  And pip\_link.link\_name = plink.link\_name  And rownum < 2  If the above returned data, then continue to the below step:  Set ucoeIdentifier = pserver.hostname  Set ucpeManagedBy = Pserver.management\_option  End If |
| **Retrieved fields:**  pip\_link2.pnf\_name | STEP NOD-100-B  A&AI Tables:  aai\_pserver pserver,  aai\_p\_interface pinterface1,  aai\_p\_interface pinterface2,  aai\_p\_interface\_physical\_link pip\_link1,  aai\_p\_interface\_physical\_link pip\_link2 | If STEP NOD-100-A returned data, Then  Determine whether this is a uCPE behind a Real Router:  Input: <ucpeHostname> from STEP NOD-100-A  Search path: pnf > p\_interface > physical\_link > p\_interface > pserver  Conditions:  And pinterface1.hostname = <ucoeHostname>  And pserver.hostname = pinterface1.hostname  And pinterface1.hostname = pip\_link1.hostname  And pinterface1.interface\_name = pip\_link1.interface\_name  And pip\_link1.link\_name = pip\_link2.link\_name  And pinterface2.interface\_name = pip\_link2.interface\_name  And pinterface2.pnf\_name = pip\_link2.pnf\_name |
| **Retrieved fields:**  vnf.vnf\_id, vnf.vnf\_type – not for output  isCascadedUCpe, | STEP NOD-100-C  A&AI Tables:  aai\_generic\_vnf vnf,  aai\_generic\_vnf\_vserver vnfVserver,  aai\_vserver\_pserver vPserver | <301033> Determine this uCPE has firewall (FW) but no vRouter (RT)  Input: <ucpeHostname> from STEP NOD-100-A  Search path: pserver > vserver> generic\_vnf  Conditions:  And vPserver.hostname = <ucpeHostname>  And vPserver.vserver\_id = vnfVserver.vserver\_id  And vnfVserver.vnf\_id = vnf.vnf\_id  And vnf.vnf\_type IN ('RT', 'FW')  If the above step returns only vnf\_type=’FW’, then  set isCascadedUCpe =Y  Else (no data returned, or RT is returned as well)  set isCascadedUCpe =N  End If |

<301033> Remove logic using tables from A&AI batch feeds.

|  |  |
| --- | --- |
| **Steps** | **Logic and Description** |
| ~~<287954a>~~  ~~isCascadedUCpe,~~  ~~ucpeIdentifier,~~  ~~<289116>~~  ~~ucpeMamagedBy~~ | ~~Access A&AI data:~~  ~~Derive isCascatedUcpe field – After the access circuit is retrieved.~~  ~~A&AI has equip\_type of ‘JUNIPER UCPE’ for uCPE.~~  ~~Input: <custAccessCktId>, retrieved from step 1.~~  ~~Output: isCascadedUCpe – derived~~  ~~ucpeIdentifier - this is the CC Host name of the uCPE~~  ~~Select Unique~~  ~~d.equipment\_name as ucpeIdentifier, -- this is the CC Host name of the uCPE~~  ~~d.managed\_by AS ucpeManagedBy <289116.141314-US640793>~~  ~~From~~  ~~INSTAR\_staging.instar\_aai\_interface i,~~  ~~INSTAR\_staging.instar\_aai\_device d~~  ~~Where 1=1~~  ~~And i.cust\_access\_ckt = <custAccessCktId>~~  ~~And d.equip\_type = ‘JUNIPER UCPE’~~  ~~And d.ptnii\_name = i.ptnii\_name~~  ~~And rownum < 2~~  ~~Set isCascadedUCpe =’N’~~  ~~Set ucoeIdentifier = null~~  ~~If the above SQL returns data, then~~  ~~Set isCascadedUCpe =Y~~  ~~Set ucoeIdentifier = d.equipment\_name~~  ~~Set ucpeManagedBy = d.managed\_by retrieved above <289116.141314-US640793>~~  ~~End If~~  ~~Populate the field with retrieved data.~~  ~~Note:~~  ~~This logic may need modification, if the circuit is modeled on the real router.~~  ~~Pending input from AAI~~ |

**297100a(1804 Rel):Data Access Logic :**

|  |  |
| --- | --- |
| **Steps,**  **Output Fields** | **Data Retrieval Logic** |
| *vpePtniiName* | For givenCR IP address(v4 ip address) execute below logic to retrieve vpe PTNII Name  INSTAR.equip\_model.ptnii\_equip\_code:  me6 – Juniper VMX Router  mee – DriveNet vPE  Select Unique  equip.ptnii\_equip\_name vpePtniiName  From  equipment equip,  equip\_model,  ip\_port\_asgmt,  ip\_assigned\_endpt,  SERIAL\_IP\_ADDR  Where 1=1  And (equip\_model.ptnii\_equip\_code in (‘~~me6’,~~'mee'))  And equip.equip\_mod\_id = equip\_model.equip\_mod\_id  And equip.equip\_id = ip\_assigned\_endpt.equip\_id  And ip\_assigned\_endpt.endpoint\_id = ip\_port\_asgmt.ip\_endpt  And ip\_port\_asgmt.cr\_addr\_id = serial\_ip\_addr.serial\_ip\_addr\_id  And serial\_ip\_addr.ip\_address=<input.CRIPaddress>  If input is v6 ipaddress then use below tables and conditions  Select Unique  equip.ptnii\_equip\_name vpePtniiName  From  equipment equip,  equip\_model,  ip\_port\_asgmt,  ip\_assigned\_endpt,  IPV6\_ASSIGNED\_LINK\_IPS,  IPV6\_PORT\_ASGMT\_MAP,  Where 1=1  And (equip\_model.ptnii\_equip\_code in (~~‘me6’~~,'mee'))  And equip.equip\_mod\_id = equip\_model.equip\_mod\_id  And equip.equip\_id = ip\_assigned\_endpt.equip\_id  And ip\_assigned\_endpt.endpoint\_id = ip\_port\_asgmt.ip\_endpt  And IPV6\_PORT\_ASGMT\_MAP.sdid = IP\_PORT\_ASGMT.sdid  And IPV6\_ASSIGNED\_LINK\_IPS.ipv6\_link\_ip\_id=  IPV6\_PORT\_ASGMT\_MAP. ipv6\_link\_ip\_id  And (IPV6\_ASSIGNED\_LINK\_IPS.ipv6\_address\_compress = CRipv6\_address input from AOTS  OR  IPV6\_ASSIGNED\_LINK\_IPS.ipv6\_address= CRipv6\_address input from AOTS  ) |
| *MISOnVPEIndicator* | Use below logic to determine the MIS on VPE Flag  select  equip.ptnii\_equip\_name vpePtniiName  service.serv\_name  From  equipment equip,  equip\_model,  ip\_port\_asgmt,  ip\_assigned\_endpt,  SERIAL\_IP\_ADDR,  service,  ip\_serv\_acc\_pt  Where 1=1  And (equip\_model.ptnii\_equip\_code in (~~'me6'~~,'mee'))  And equip.equip\_mod\_id = equip\_model.equip\_mod\_id  And equip.equip\_id = ip\_assigned\_endpt.equip\_id  And ip\_assigned\_endpt.endpoint\_id = ip\_port\_asgmt.ip\_endpt  And ip\_port\_asgmt.ar\_addr\_id = serial\_ip\_addr.serial\_ip\_addr\_id  and ip\_port\_asgmt.serv\_acc\_pt\_id=ip\_serv\_acc\_pt.serv\_acc\_pt\_id  and ip\_serv\_acc\_pt.serv\_id=service .serv\_id  and service.serv\_name='MIS'  and equip.ptnii\_equip\_name=<vpePtniiname from above logic>  If serv\_name =’MIS’ then set misOnVpeFlag=’True’ else set it to False. |

**<270843d>**

**Data Access Logic:**

* Retrieving from INSTAR: Physical PAID
* Call INSTAR WS getLayer3Data (physicalPAID) and parsing result
* Retrieve from NISE-CORE: Tie-down and LEC circuit, vendor

|  |  |
| --- | --- |
| **Steps,**  **Output Fields** | **Data Retrieval Logic** |
| ~~Step 200~~  ~~physicalPAID,~~  ~~interfaceString~~ | **<270843d-upd1>**  **Logical PAID should be used to trigger the INSTAR WS in order to get detaild data. So to remove the logic retrieving physical PAID.**  **~~Retrieve Physical PAID and Physical Interface String~~**  ~~Input: <acc\_ckt> - customer access circuit, retrieved from earlier steps.~~  ~~Data Source: INSTAR MV~~  ~~SELECT~~  ~~pa.ip\_port\_asgmt\_id AS physicalPAID,~~  ~~pa.interface\_string AS interfaceString~~  ~~FROM~~  ~~ip\_port\_asgmt pa,~~  ~~ip\_serv\_acc\_pt sap,~~  ~~cust\_access ca~~  ~~Where 1=1~~  ~~And ca.acc\_ckt = <acc\_ckt>~~  ~~And ca.site\_id = sap.site\_id~~  ~~And sap.serv\_acc\_pt\_id = pa.serv\_acc\_pt\_id~~  ~~And sap.sap\_type = 'Physical'~~  **~~Populate interfaceString field.~~** |
| Step 210 | **<270843d>**  **Call INSTAR WS getLayer3Data (physicalPAID) and parsing result (Check attached INSTAR AID)**  **Input:** ~~<physicalPAID> - physical PAID retrieved from Step 200.~~  <logicalPAID> - logical PAID retrieved from Step 1. <270843d-upd1>  **Call INSTAR WS.getLayer3Data (<PAID>);**  **Parsing the output to populate <VlanLayer3Data>: From the getLayer3Data response –**  **Retrieve content in ClassOfService structure, populate <ClassOfService>**   * getLayer3Data.ClassOfService.IngressProfileId -> <ClassOfService>.ingressProfileId * getLayer3Data.ClassOfService.EgressProfileId -> <ClassOfService>.egressProfileId * getLayer3Data.ClassOfService.CosPackage -> <ClassOfService>.cosPackage * getLayer3Data.ClassOfService.CosPolicyType -> <ClassOfService>.cosPolicyType * getLayer3Data.ClassOfService.DefaultTunnelClassIndicator -> <ClassOfService>.defaultTunnelClassIndicato   **Retrieve content in BgpProtocolData structure, populate <BgpProtocolData>**   * getLayer3Data.BgpProtocolData.BgpType -> <BgpProtocolData>.bgpType * getLayer3Data.BgpProtocolData.RoutingProtocol -> <BgpProtocolData>.routingProtocol * Derive <BgpProtocolData>.isMultiBfd - bfdIndicator, by checking MisBgpParameter.BfdHeartBeat below.   Values:  Y - If MisBgpParameter.BfdHeartBeat populated  N - otherwise   * getLayer3Data.BgpProtocolData.AclCategory -> <BgpProtocolData>.packetFilterType * **Retrieve content in MisBgpParameter structure, populate <MisBgpParameter>** * getLayer3Data.BgpProtocolData.MisBgpParameter.RouterFilter -> <MisBgpParameter>.routerFilter * getLayer3Data.BgpProtocolData.MisBgpParameter.BfdHeartBeat -> <MisBgpParameter>.bfdInterval   Set <BgpProtocolData>.bfdIndicator with values:  Y - If this value populated  N - otherwise   * getLayer3Data.BgpProtocolData.MisBgpParameter.NeighborAddress.ipAddress -> <MisBgpParameter>.neighborAddress   **Retrieve content in Layer3Info structure, populate vlanLogicalInterfaceString**   * getLayer3Data.Layer3Info.InterfaceString -> <VlanLayer3Data>.vlanLogicalInterfaceString   **Note:** <270843d-upd2>   * As of 8/23/2016, INSTAR AID has Layer3Info with occurrence 1-1, but WSDL has 1-N. * This field will stay unchanged (matching Instar AID); * The code will pick up the first Layer3Info instance and get the InterfaceString value.   **Derive isMultiBgp:**  If there are more than one BgpProtocolData occurrence, then  Set isMultiBgp = ‘Y’  Else  Set isMultiBgp = ‘N’  End If  **Derive isMultiWan:**  If there are more than one "ns4:WanIPAddress" occurrences in InterfaceAddress sessions, then  Set isMultiWan = ‘Y’  Else  Set isMultiWan = ‘N’  End If |
| Step 220 | **<270843d>**  **Retrieve NISE-CORE to get, using the CLCI of the ACCESS\_CKT:**  Layer 1 Tie Down   (source NISE\_C) (will be Working Tie Down and Protection Tie Down)   * Location CLLI – from CHF\_CIRCUIT\_TERM\_A or CHF\_CIRCUIT\_TERM\_Z whichever has the Tie Down * Working  TD FIC Code with panel and port – WKG\_TD  (CHF\_TDI\_WORKING\_A) or (CHF\_TDI\_WORKING\_Z) * Protection TD  FIC Code with panel and port - PROT\_TD  (CHF\_TDI\_PROTECTION\_A) or (CHF\_TDI\_PROTECTION\_Z) * LEC\_CKT\_ID * ACCESS\_VENDOR   Type of Circuit – OTN, ETH, TDM  **Input:** <custAccessCircuit> - custAccessCircuitId from earlier step |
| Step 220.1 | **Covert <custAccessCircuit> into CLCI format <clci-cust\_accessCircuit>**  **Input:** <custAccessCircuit> - custAccessCircuitId from earlier step.  Output: clci-custAccessCircuit  **Call common logic** CircuitFormatConversion\_ICORE\_to\_CLCI\_forNise (<custAccessCircuit>)  Get : clciI-custAccessCircuit |
| Step 220.2 | **Retrieve NISE-CORE to get Tie Down**  Input: <clci-custAccessCircuit>.  Output: LocationClli, WorkingTieDown, ProtectionTieDown, LecCircuit, accessVendor;  Data Source: NISE-CORE  **Select**  chf\_app\_circuit\_nm AS circuitID,  --get LocationClli based on TieDown available at A or Z end  (Case  When Chf\_tdi\_working\_a is not null Then Chf\_circuit\_term\_a  Else Chf\_circuit\_term\_z  End  ) locationClli,  NVL (Chf\_tdi\_working\_a, Chf\_tdi\_working\_z) AS workingTieDown,  NVL (Chf\_tdi\_protection\_a, Chf\_tdi\_protection\_z) AS protectionTieDown,  --Derive Circuit Type  Decode (substr(chf\_app\_circuit\_nm, inStr(chf\_app\_circuit\_nm, '/', 1, 1)+1, 2),  'DH', 'TDM',  'IW', 'TDM',  'IV', 'TDM',  'AG', 'TDM',  'DN', 'TDM',  'UW', 'OTN',  'UX', 'OTN',  'ETH') circuitType,  --Derive/construct LEC Circuit ID  --Construct it using the subfields for A or Z end, depending on the data is available, and it’s not AT&T  Case  when (length ( --construct LEC-A circuit  CHF\_LEC\_CKT\_ID\_A\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SVC ||  CHF\_LEC\_CKT\_ID\_A\_JUR ||  CHF\_LEC\_CKT\_ID\_A\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SEG)  ) > 5  And chf\_access\_vendor\_a IS NOT NULL And Upper(chf\_access\_vendor\_a) != 'AT&T'  Then (  CHF\_LEC\_CKT\_ID\_A\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SVC ||  CHF\_LEC\_CKT\_ID\_A\_JUR ||  CHF\_LEC\_CKT\_ID\_A\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SEG)  when (length ( --construct LEC-Z circuit  CHF\_LEC\_CKT\_ID\_Z\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SVC ||  CHF\_LEC\_CKT\_ID\_Z\_JUR ||  CHF\_LEC\_CKT\_ID\_Z\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SEG)  ) > 5  And chf\_access\_vendor\_z IS NOT NULL And Upper(chf\_access\_vendor\_z) != 'AT&T'  Then (  CHF\_LEC\_CKT\_ID\_Z\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SVC ||  CHF\_LEC\_CKT\_ID\_Z\_JUR ||  CHF\_LEC\_CKT\_ID\_Z\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SEG)  when (length ( --get LEC-A circuit  CHF\_LEC\_CKT\_ID\_A\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SVC ||  CHF\_LEC\_CKT\_ID\_A\_JUR ||  CHF\_LEC\_CKT\_ID\_A\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SEG)  ) = 5  And Length (CHF\_ALT\_LEC\_CKT\_ID\_A) > 0  And chf\_access\_vendor\_a IS NOT NULL And Upper(chf\_access\_vendor\_a) != 'AT&T'  Then CHF\_ALT\_LEC\_CKT\_ID\_A    when (length ( --get LEC-Z circuit  CHF\_LEC\_CKT\_ID\_Z\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SVC ||  CHF\_LEC\_CKT\_ID\_Z\_JUR ||  CHF\_LEC\_CKT\_ID\_Z\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SEG)  ) = 5  And Length (CHF\_ALT\_LEC\_CKT\_ID\_Z) > 0  And chf\_access\_vendor\_z IS NOT NULL And Upper(chf\_access\_vendor\_z) != 'AT&T'  Then CHF\_ALT\_LEC\_CKT\_ID\_Z  end lecCktId,  --Get Access Vendor  --Check it using the subfields for A or Z end, depending on the data is available, and it’s not AT&T  Case  when (length (  CHF\_LEC\_CKT\_ID\_A\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SVC ||  CHF\_LEC\_CKT\_ID\_A\_JUR ||  CHF\_LEC\_CKT\_ID\_A\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SEG)  ) > 5  And chf\_access\_vendor\_a IS NOT NULL And Upper(chf\_access\_vendor\_a) != 'AT&T'  Then CHF\_ACCESS\_VENDOR\_A  when (length (  CHF\_LEC\_CKT\_ID\_Z\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SVC ||  CHF\_LEC\_CKT\_ID\_Z\_JUR ||  CHF\_LEC\_CKT\_ID\_Z\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SEG)  ) > 5  And chf\_access\_vendor\_z IS NOT NULL And Upper(chf\_access\_vendor\_z) != 'AT&T'  Then CHF\_ACCESS\_VENDOR\_Z  when (length (  CHF\_LEC\_CKT\_ID\_A\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SVC ||  CHF\_LEC\_CKT\_ID\_A\_JUR ||  CHF\_LEC\_CKT\_ID\_A\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_A\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_A\_SEG)  ) = 5  And Length(CHF\_ALT\_LEC\_CKT\_ID\_A) > 0  And chf\_access\_vendor\_a IS NOT NULL And Upper(chf\_access\_vendor\_a) != 'AT&T'  Then CHF\_ACCESS\_VENDOR\_A  when (length (  CHF\_LEC\_CKT\_ID\_Z\_PREFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SVC ||  CHF\_LEC\_CKT\_ID\_Z\_JUR ||  CHF\_LEC\_CKT\_ID\_Z\_MOD || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SERIAL || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SUFFIX || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_COMP || '/' ||  CHF\_LEC\_CKT\_ID\_Z\_SEG)  ) = 5  And length (CHF\_ALT\_LEC\_CKT\_ID\_Z) > 0  And chf\_access\_vendor\_z IS NOT NULL And Upper(chf\_access\_vendor\_z) != 'AT&T'  Then CHF\_ACCESS\_VENDOR\_Z  End accessVendor--,  ~~--CHF\_ALT\_LEC\_CKT\_ID\_A AS altLecCktIdA,~~  ~~--CHF\_ALT\_LEC\_CKT\_ID\_Z AS altLecCktIdZ,~~  ~~--CHF\_ACCESS\_VENDOR\_A AS accessVendorA,~~  ~~--CHF\_ACCESS\_VENDOR\_Z AS accessVendorZ~~  **From circuit\_has\_fact**  **Where 1=1**  And chf\_app\_circuit\_nm = <input-clci-custAccessCircuit>  ~~and (CHF\_ALT\_LEC\_CKT\_ID\_A is not null or CHF\_ALT\_LEC\_CKT\_ID\_Z is not null)~~  ~~And rownum<50~~ |
|  |  |

**<289207>**

**STEP-300 Data Access Logic: Populating <NxXGigEBundleList>**

* Retrieving the IOC1 circuits for the NxXGigE bundle
* Retrieve from NISE-CORE: Tie-down and LEC circuit, vendor

|  |  |
| --- | --- |
| **Steps,**  **Output Fields** | **Data Retrieval Logic** |
| Step 300  <NxXGigEBundleList>  ioc1CktId  <Layer1TieDown>  </NxXGigEBundleList> | **<289207>**  **For LAG\_TYPE with values shown below, retrieve the IOC1 circuits in the bundle:**  Nx10GigE  Nx100GigE  Input: <lag\_type>, <lag\_sdid> --From Step 1  Data Source: INSTAR MV  Set gigEthernetLagType = lag\_type value  Follow Step 300, with Ip\_serv\_acc\_pt.lag\_sdid  **Step-300-A: Retrieve IOC1 circuits**  <289207-QC176237>  UAT issue with empty ioc1 field in <NxXGigEBundleList>.  Fix: Use acc\_ckt field when ioc1 is blank – use NVL to do this.  ~~Select Unique ca.ioc1~~  Select Unique  NVL (ca.ioc1, ca.acc\_ckt) AS ioc1 <289207-QC176237>  Equipment.ptnii\_equip\_name, ~~<289207-CR164691>~~ <289503 CR165525>  Component.beg\_slot AS slot,  Component.port,  Round((Component.port / 1000) -1) AS baseport  From  cust\_access ca,  ip\_serv\_acc\_pt sap,  ip\_port\_asgmt~~<289207-CR164691>~~ <289503 CR165525>  ip\_assigned\_endpt,  component,  equipment  Where 1=1  And sap.lag\_sdid = <lag\_sdid>  And sap.lag\_type = <lag\_type>  And sap.site\_id = ca.site\_id  And ip\_port\_asgmt.Serv\_acc\_pt\_id = sap.Serv\_acc\_pt\_id ~~<289207-CR164691>~~ <289503 CR165525>  And ip\_port\_asgmt.ip\_endpt = ip\_assigned\_endpt.endpoint\_id  And component.comp\_id = ip\_assigned\_endpt.comp\_id  And equipment.equip\_id = component.equip\_id  **Populate the ~~ico1~~ retrieved records to ~~ioc1CktId~~ fields in NxXGigEBundleList structure.**  **Step-300-B: Retrieve Tie down info for each IOC1 circuit**  Follow Step 220, with the IOC1 circuits as input to that step.  Populate the <Layer1TieDown> structure for each IOC1 circuit.  Sample SQL (this should NOT be used in the implementation as exact reflection of the logic):  Select Unique  NVL (ca.ioc1, ca.acc\_ckt) AS ioc1,  Equipment.ptnii\_equip\_name,  Component.beg\_slot,  Component.port,  round((Component.port / 1000) -1) baseport  From  cust\_access ca,  ip\_serv\_acc\_pt sap,  ip\_port\_asgmt,  ip\_assigned\_endpt,  component,  equipment  Where 1=1  And sap.lag\_sdid is not null  And sap.site\_id = ca.site\_id  And ip\_port\_asgmt.Serv\_acc\_pt\_id = sap.Serv\_acc\_pt\_id  And ip\_port\_asgmt.ip\_endpt = ip\_assigned\_endpt.endpoint\_id  And Component.comp\_id = ip\_assigned\_endpt.comp\_id  And equipment.equip\_id = component.equip\_id  And sap.lag\_type in ('Nx10GigE','Nx100GigE')  And rownum<40  Order by ptnii\_equip\_name   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **IOC1** | **PTNII\_EQUIP\_NAME** | **BEG\_SLOT** | **PORT** | **BASEPORT** | | ABTD.170139.245.ATI | cgcil52me9 | 6 | 3000 | 2 | | ABTD.170139.246.ATI | cgcil52me9 | 7 | 1000 | 0 | | ABTD.170139.247.ATI | cgcil52me9 | 7 | 3000 | 2 | | BHEC.170298.101.ATI | cgcil52me9 | 3 | 1000 | 0 | | BHEC.170298.201.ATI | cgcil52me9 | 3 | 2000 | 1 | | QWVR.170240.355.ATI | cgcil52me9 | 6 | 4000 | 3 | | QWVR.170240.356.ATI | cgcil52me9 | 7 | 2000 | 1 | | QWVR.170240.357.ATI | cgcil52me9 | 7 | 4000 | 3 | | ZLEC.006496..ATI | cgcil52me9 | 5 | 1000 | 0 | | ZLEC.048919..ATI | cgcil52me9 | 4 | 2000 | 1 | | ZLEC.128971..ATI | cgcil52me9 | 5 | 2000 | 1 | | ZLEC.131327..ATI | cgcil52me9 | 4 | 3000 | 2 | | ZLEC.371658..ATI | cgcil52me9 | 5 | 4000 | 3 | | ZLEC.770484..ATI | cgcil52me9 | 5 | 3000 | 2 | | ZLEC.852682..ATI | cgcil52me9 | 6 | 2000 | 1 | | ZLEC.944240..ATI | cgcil52me9 | 6 | 1000 | 0 | | ZLEC.948854..ATI | cgcil52me9 | 4 | 1000 | 0 | | ZLEC.996517..ATI | cgcil52me9 | 4 | 4000 | 3 | | L8YX.146737..ATI | cgcil62me9 | 5 | 1000 | 0 | | L8YX.146738..ATI | cgcil62me9 | 5 | 4000 | 3 | | DNEC.146718.037.ATI | cgcil63me9 | 5 | 4000 | 3 | | DNEC.146718.038.ATI | cgcil63me9 | 5 | 1000 | 0 | | DNEC.146720.037.ATI | cgcil63me9 | 5 | 2000 | 1 | | DNEC.146720.038.ATI | cgcil63me9 | 5 | 3000 | 2 | | DNEC.146721.037.ATI | cgcil64me9 | 5 | 4000 | 3 | | DNEC.146721.038.ATI | cgcil64me9 | 5 | 1000 | 0 | | /GAYA/160485/ATI | cgcil67me9 | 0 | 2000 | 1 | | /GAYA/160486/ATI | cgcil67me9 | 2 | 4000 | 3 | | /GAYU/160485/ATI | cgcil67me9 | 0 | 1000 | 0 | | /GAYU/160486/ATI | cgcil67me9 | 2 | 3000 | 2 | | ABRD.988882.223.ATI | cgcil67me9 | 2 | 1000 | 0 | | QWGR.194002.365.ATI | cgcil67me9 | 2 | 2000 | 1 | | ABTD.170139.249.ATI | cgcil68me9 | 0 | 1000 | 0 | | QWVR.170240.359.ATI | cgcil68me9 | 0 | 2000 | 1 | | /PSPO/354918/ATI | n54ny46me9 | 0 | 1000 | 0 | | /PSPO/354919/ATI | n54ny46me9 | 0 | 3000 | 2 | | /PSPO/454918/ATI | n54ny46me9 | 0 | 2000 | 1 | | /PSPO/454919/ATI | n54ny46me9 | 0 | 4000 | 3 | | PSPO.354920..ATI | n54ny46me9 | 1 | 1000 | 0 | |

Existing production SQLs (As of July 2016)

|  |
| --- |
| 2. Clarify query:  SELECT  OutOfBandPotsLine, MISLanPortIP, LanPortIPv6, DeviceModel, CSUModel, PrimaryDNSProvider, SecondaryDNSProvider,  RouterIOS, RouterFeatures,  AccessRouterName,  restrictionIndicator, ipaddress as CustRouterIpaddress  FROM IPRouterData where ipaddress = '<input>'; |
| 2. NC3 query: Select distinct lanIP.ip\_address as misLanPortIP, lanIP.ipv6\_address as misLanPortIPv6, main\_eqp.common\_name as pdnsProvider, er.model as deviceModel, main\_eqp.firmware\_version as routerIOS, er.description as routerFeatures, parm.value as OutOfBandPotsLine  From  nc3\_mat.ip\_interface ip, nc3\_mat.ip\_interface lanIP, nc3\_mat.log\_port lp,  nc3\_mat.port p, nc3\_mat.eqp e, nc3\_mat.eqp main\_eqp, nc3\_mat.eqp\_ref er,  nc3\_mat.parm parm  Where lp.odbid = ip.odbid\_log\_port  and p.odbid = lp.odbid\_port  and e.odbid = p.odbid\_eqp  and main\_eqp.odbid\_eqp\_ref = er.odbid  and e.odbid\_main\_eqp = main\_eqp.odbid  and parm.odbid\_cmponent = main\_eqp.odbid  and parm.internal\_label = 'DIAL\_NUMBER'  and lanIP.odbid in  (select  ip\_interface.odbid  from nc3\_mat.ip\_interface , nc3\_mat.log\_port, nc3\_mat.log\_port\_ref, nc3\_mat.port, nc3\_mat.eqp  card, nc3\_mat.eqp  equip  where log\_port.odbid = ip\_interface.odbid\_log\_port  and port.odbid = log\_port.odbid\_port  and card.odbid = port.odbid\_eqp  and card.odbid\_main\_eqp = equip.odbid  and equip.odbid = main\_eqp.odbid  and log\_port\_ref.odbid = log\_port.odbid\_log\_port\_ref  and log\_port\_ref.short\_name = 'LAN-INTERFACE')  and e.status in ('Useable')  and e.stop\_date > sysdate  and parm.status in ('Started','Planned Stop','Useable','Active')  and p.stop\_date > sysdate  and lanIP.stop\_date > sysdate  and lanIP.status in ('Started','Planned Stop','Useable','Active')  and (trim(ip.ip\_address) = '<input>'  OR  trim(ip.ipv6\_address) = '<input>'); |
| 3. Instar query: select ipa.IP\_PORT\_ASGMT\_ID  from ip\_port\_asgmt ipa,  serial\_ip\_addr sia  where sia.ip\_address= <input>  and ipa.wan\_addr\_id = sia.serial\_ip\_addr\_id  Union  Select ipa.IP\_PORT\_ASGMT\_ID  From ip\_port\_asgmt ipa,  serial\_ip\_addr sia  where sia.ip\_address= <input>  and ipa.cr\_addr\_id = sia.serial\_ip\_addr\_id  union  select ipa.IP\_PORT\_ASGMT\_ID  from ip\_port\_asgmt ipa,  serial\_ip\_addr sia  where sia.ip\_address= <input>  and ipa.ar\_addr\_id = sia.serial\_ip\_addr\_id |
| 4. USRP query: Select Distinct  Decode (iab.prdns\_provider, 0, 'AT&T', 1, 'Customer', 2, 'Other')  AS pdnsProvider,  Decode (iab.sdns\_provider, 0, 'AT&T', 1, 'Customer', 2, 'Other')  AS sdnsProvider,  ic.pots\_number  AS outOfBandPotsLine,  ic.csu\_dsu\_model  AS csuModel,  substr(cic.mis\_ar\_name, 1, 40)  AS AccessRouterName,  cic.equip\_model  AS DeviceModel  From  ip\_address\_block iab,  ip\_cpe ic,  connection\_ip\_ckt cic,  customer\_location l,  ip\_info ii,  db\_const\_def def  Where (1=1)  And ii.log\_ip\_id = cic.log\_ip\_id  And iab.logical\_ip\_id = ii.log\_ip\_id  And ic.logical\_ip\_id = ii.log\_ip\_id  And l.cust\_location\_id = ii.cust\_location\_id  And (cic.wan\_link\_address = '<input>'  Or cic.wan\_link\_ipv6\_address = '<input>'  Or cic.wan\_link\_ipv6\_compr\_address = '<input>'  Or cic.cr\_serial\_int\_ipv6\_addr = '<input>'  Or cic.cr\_serial\_int\_ipv6\_compr\_addr = '<input>'  or cic.IPD\_PORT\_ASGMT\_ID= '<value from query 3>')  Order by ic.pots\_number  Note at least one of query 1, 2, or 4 must return data in order for getCPEquipmentDetailsByWANLinkIPAddr to return a successful response. |

### Common logic – convert circuit format: CircuitFormatConversion\_ICORE\_to\_CLCI\_forNise – (created for 286284, with small change)

|  |  |
| --- | --- |
| **Steps** | **Logic and Description** |
|  | <286284> **Reformat input Circuit in ICORE format to CLCI format.**  Input: input cktID - ICORE format  Output: CLCI-CircuitID  Sample values in ICORE format:   |  | | --- | | INSTAR.cust\_access.acc\_ckt | | 12345678901234567890123  cktType = "NoPREFIX"  DHEC.446925..ATI  DHEC.348546.802.ATI  cktType = "PREFIX"  02.DHEC.446925..ATI  02.DHEC.348546.802.ATI |   CLCI format:  /DHEC/956448/807/ATI/  // Positions are referenced in parenthesis e.g. (3) is position 3 in the input circuit ID.  Set PREFIX = input cktID (1-2) or ‘‘  If input cktID(3) is not ‘.’  PREFIX = ‘‘ (0 blank space)  cktType = "NoPREFIX"  ELSE  PREFIX = input cktID(1-2)  cktType = "PREFIX"  // The following steps assume that either the PREFIX was included in the ICORE Circuit ID or it wasn't included. Thus, two sets of character positions are mentioned.  Set SVCCD = input cktID(4-5) or (1-2)  If cktType = "PREFIX"  SVCCD = input cktID (4-5)  Else  SVCCD = input cktID (1-2)  Set SVCMOD = input cktID(6-7) or (3-4)  If cktType = "PREFIX"  SVCMOD = input cktID (6-7)  Else  SVCMOD = input cktID (3-4)  Set SERNO = input cktID (9-14) or (6-11)  If cktType = "PREFIX"  SERNO = input cktID (9-14)  Else  SERNO = input cktID (6-11)  Set SUFFIX = input cktID (16-18) or (13-15)  If cktType = "PREFIX"  SUFFIX = input cktID (16-18)  If "." in the value then  SUFFIX = ‘’ (0 blank space)  End if  Else if cktType = "NoPREFIX"  SUFFIX = input cktID (13-15)  If "." in the value then  SUFFIX = ‘’ (0 blank space)  End if  End If  cktType = "NoPREFIX"  DHEC.446925..ATI  DHEC.348546.802.ATI  cktType = "PREFIX"  02.DHEC.446925..ATI  02.DHEC.348546.802.ATI  12345678901234567890123  Set CO = input cktID (20-23), or (17-20), or (14-17)  If cktType = "NoPREFIX"  If SUFFIX = ‘’ (0 blank space) then  CO = input cktID (14-17)  Else  CO = input cktID (17-20)  End if  Else if cktType = "PREFIX"  If SUFFIX = ‘’ (0 blank space) then  CO = input cktID (17-20)  Else  CO = input cktID (20-23)  End if  End If  If PREFIX exist,  Set CLCI CircuitID = ‘/’ + PREFIX + ‘/’ + SVCCD + SVCMOD + ‘/’ + SERNO + ‘/’ + SUFFIX + ‘/’ + Trim(CO) + ‘/’  Else  Set CLCI CircuitID = ‘/’ + SVCCD + SVCMOD + ‘/’ + SERNO + ‘/’ + SUFFIX + ‘/’ + Trim(CO) + ‘/’  End If |
|  |  |



## Alternative Designs

N/A.

No alternative designs will be considered beyond the design already in production.

## Assumptions/Risks

N/A – Nothing changed here

## Traceability Matrix

|  |  |
| --- | --- |
| Requirement ID | Design Element Identifier |
|  |  |

## Pre-Production Disaster Recovery Planning

N/A – Nothing changed here

## Other Plans and References

N/A – Nothing changed here

|  |
| --- |
|  |

## 

## Acceptance & Approvals

Overview

The Approvers of this work product agree that this document is acceptable and complete to the best of their knowledge and will be used by the project team as an official deliverable for the project. It is further agreed that this document can now be baselined and any changes to these sections from this point forward must follow the Managing Change in the IT UP.

Embed evidence of approval in the review table below, or use the PRISM Approval Functionality in the Project Workflow Module Workflow Template View.

Approvers

|  |  |  |  |
| --- | --- | --- | --- |
| **ATTUID and Name** | **Role** | **Group/Application** | **Version Approved, Approval Date and Approval Evidence** |
| mh6892 – Ming Ho | Artifact Owner - Application Architect | GCP | see PRISM for evidence of approval |
|  | Artifact Contributor | GCP  Note: xyz is approving for both the GCP Dev Mgr and GCP Test Mgr | see PRISM for evidence of approval |
|  | Lead PM |  | see PRISM for evidence of approval |
|  |  |  |  |

Appendix