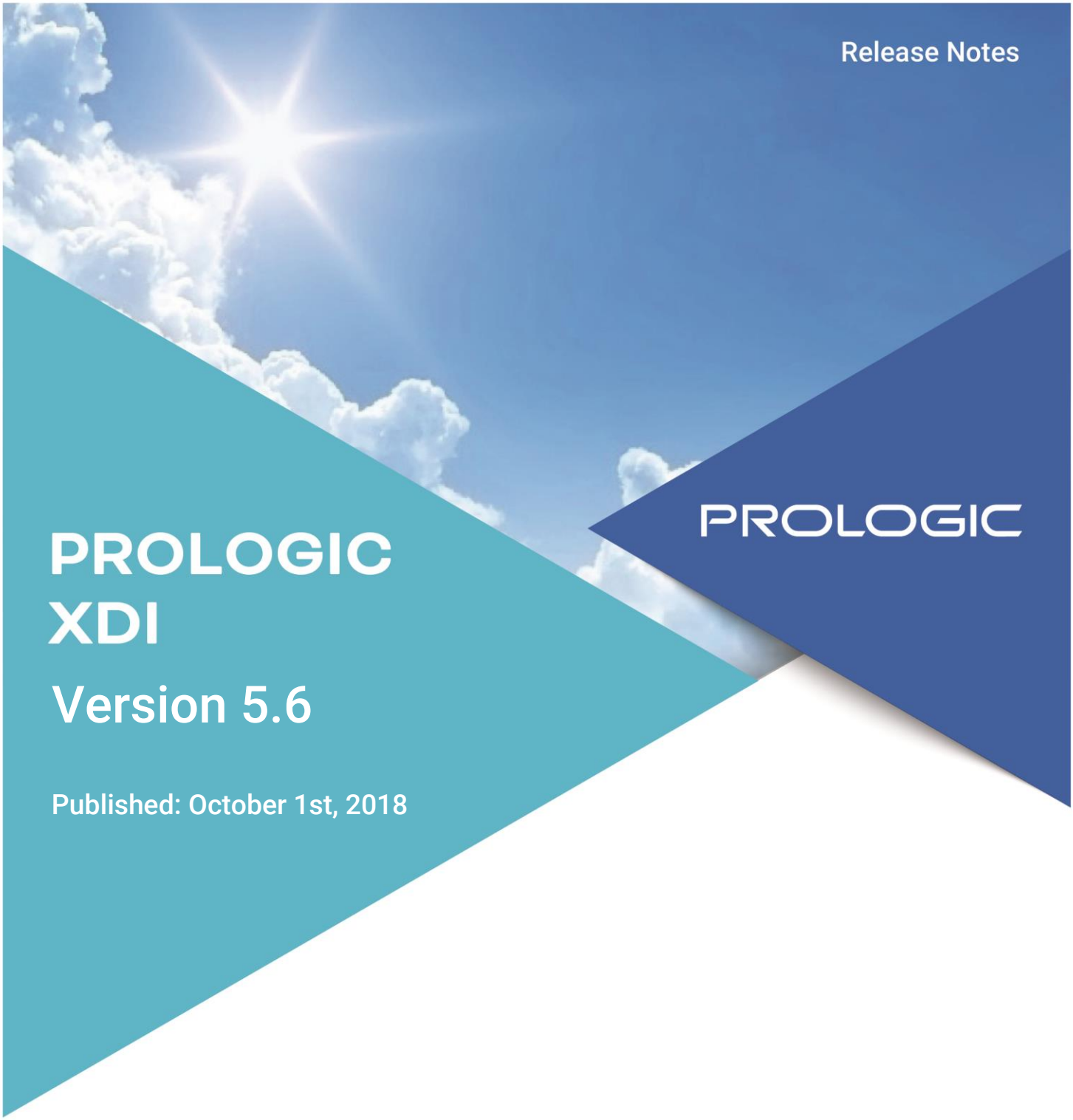


PROLOGIC XDI

Version 5.6

Published: October 1st, 2018

TABLE OF CONTENTS

1. Release Overview 3
2. Summary of Issues Fixed 3
   2.1 Issues Fixed in v5.6 3
3. Known Issues 3
4. Prologic XDI System Requirements 4
   4.1 XDI Overview 4
   4.2 XDI Explained 4
   4.3 XML Details 5
   4.4 XSD and XSLT Lookup 6
   4.5 XSD and XSLT Lookup 7
   4.6 Third Party Integration 7
5. BTe Store6 Interface 8
   5.1 Service Operations 8
   5.2 Client Operations 9
6. CIMS Based Services 11
   6.1 Feeds 11
   6.2 Operations 12
7. Upgrade Instructions 14

1. Release Overview

Prologic XDI version 5.6 release is focused on fixing issues reported by customers in previous releases as well as internally reported issues. In order to provide the highest quality in our releases, each release is tested to the best of our ability.

This release has 3 fixes implemented.

Release Version	Prologic XDI 5.6
Release Date	October 1st, 2018
Release Type	Generally Available
Deliverables	Release Notes Documentation

2. Summary of Issues Fixed

2.1 Issues Fixed in v5.6

Detailed below are the fixes in this release:

- Wishworks client was enhanced in this release to not mark CIMS queue as failure when there is no workload to process. (PROLOGICCID-4111)
- Mulesoft client program was enhanced to use database identifier when encrypting/decrypting credentials. This will avoid issues during the database refresh. (PROLOGICCID-4156)
- Import picks now defaults the cancel short pick flag to Y so that CDI processing will call the relevant routine. (PROLOGICCID-5662)

3. Known Issues

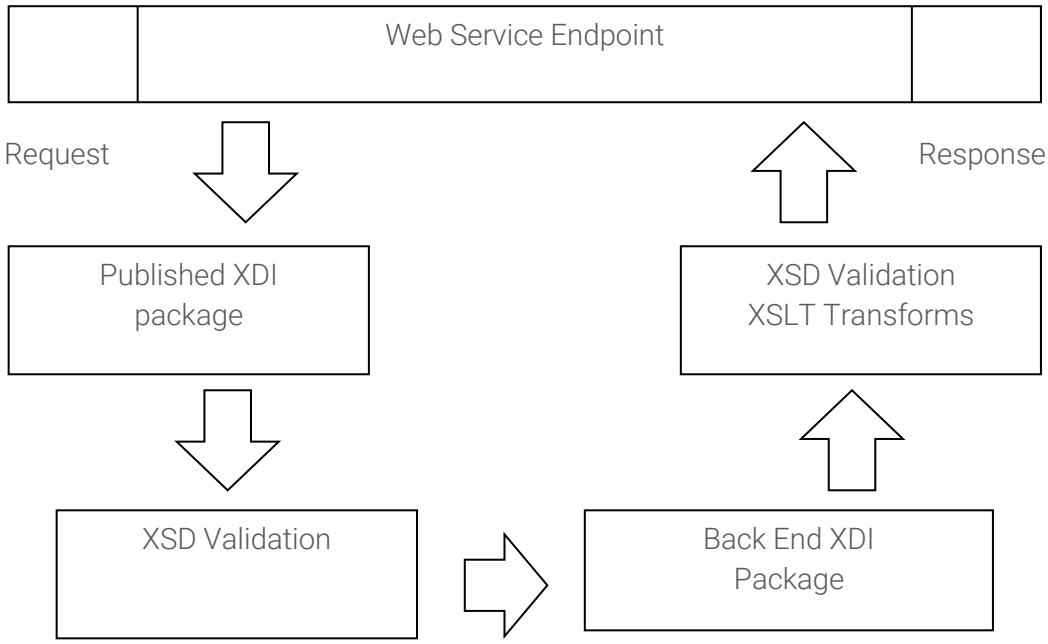
Known issues for XDI:

- Unable to process orders at the warehouse level due to duplicates in point_items_full view. (PROLOGICCID-3263)
- markProductRegister function fails with ORA-04030: out of process memory. (PROLOGICCID-6476)

4. Prologic XDI System Requirements

4.1 XDI Overview

This document explains the main design of the XDI web service operations provided by CIMS. It is an overview of the methodology involved rather than a detailed account of the actual services provided. The layers of the XDI can be represented as follows.



Here the web service endpoint is presented to clients as a URL accessible via HTTP. Using the endpoint a subscriber to the web service can call the web service operations that directly map onto to a PL/Sql package.

The package validates incoming messages taken from the HTTP request object and hands them to a back-end package (or packages) to produce output for the HTTP response object.

4.2 XDI Explained

XDI (XML services for Data Interchange) provides an open standard approach for access to data structures in CIMS.

The XDI accesses the CIMS database through a number of logical layers. At the outermost layer is the set of XDI web service endpoints. Clients connect to these endpoints (typically realised as HTTP URIs able to accept SOAP encapsulated XML messages) to send requests and receive a response.

The details of the request sent could be simply a stock look-up request or a complex XML document detailing a complete sales order. One of the key ideas behind the XDI layer is that it does not retain any permanent data.

The web service endpoint is a proxy for a J2EE published package residing in the CIMS database within the XDI schema. The package directly maps to a context root in the web server to form the endpoint. Each function in the package becomes a web service operation.
The published package is called passing in the arguments (an XML document) from the SOAP invocation, to the function handling the web service operation. The operation name is the name of the function.

The published package is typically small. Each function handles the input XML, strips out the relevant parameters and uses these to call a back-end packaged function which creates the XMLtype instance that is returned as the output to the service.

Both the input XML document and the final XML result for output are strictly validated against an XSD (XML Schema Definition). The XSD validation ensures that required components are present in the input. The validation process is also able to apply defaults to missing XML elements.

The output, having been validated to the XSD that describes the web service operation, can also (optionally) have an XSLT (XML Style sheet) applied to it to perform a transform. Using XSLT the output can be manipulated and a totally new document can be created. This would allow two different clients to subscribe to the same web service operations and receive completely different results.

There are two crucial lookup tables that allow the linking of the stored XSD and XSLT documents to be linked to version of the published and back-end PL/SQL functions that handle the requests and responses. The lookup tables ensure that the correct XSD is used for the code and the correct XSLT is applied to the output based on the client.

4.3 XML Details

The XML used in the request and response messages for the web services have wrapper elements that describe the service being called and identify the subscriber.
An input document for a service will look something like this –

```
<aServiceINPUT>
   <CALLER Id="client" Version="1.0" />
   <PARAMS>
   ...
   </PARAMS>
</aServiceINPUT>
```

And an output document will have the following form –

```
<aServiceOUTPUT
      CalledAt="Thu Jan 17 23:17:32 UTC 2008"
      Timestamp="Thu Jan 17 23:17:33 UTC 2008"
      Version="1.1.1.1">
   ...
</aServiceOUTPUT>
```

In both cases the ellipses would be replaced with the relevant input parameters needed and output details generated by the particular service being called.

The service name is prefixed to both the INPUT and OUTPUT tags. This enables more efficient coding for client applications and provides clarity.

The CALLER tag in the input document identifies the subscribing client and their version identifier. Using this information, we can tailor the output for the client by performing a lookup on a table holding this information in conjunction with an XSLT document with which a transform can be made on the output XML.

The Called_At attribute of the OUTPUT tag is the fully qualified time for the service request.
The Timestamp attribute of the OUTPUT tag is the fully qualified time of the completion of the request.
These two timestamps used together can be used to show the total elapsed time for a service operation to be executed.

The VERSION attribute of the OUTPUT tag has four components. These are the version of the function in the published package, the version of the corresponding XSD document, the version of the back-end packaged function that creates the XML output and the version of the XSD that the output conforms to.

4.4 XSD and XSLT Lookup

As mentioned earlier, various pieces of information are taken from the input XML to decide how to create the output XML.

First the CALLER tag in the input document is examined and validated against a lookup table of valid subscribers for the service operation being called. If the lookup fails then the service call is aborted. If the lookup is successful then the information returned includes an (optional) XSLT document.

The input XML must conform to the XSD for that operation. When called, the published function looks up the schema document corresponding to the function and validates the passed in XML. This has the side effect of populating any default values in the XML as well as ensuring the correct grammar.

The output function is then called and the results are passed back into the published function. It should be noted that the back-end function does not add the enclosing OUTPUT tag. This tag is added in the published function so that the back-end function is not tied to a single web service operation.

At this point the XSLT document - if needed - can then be used by the published package function to transform the output XML produced by the back-end function.

The XSD for the output is now found and the result XML validated against it. Again, the validation process can fill in any defaults specified in the XSD.

Provided that the XML has passed XSD validation the results are returned back to the calling subscriber. If at any point an error has been raised then the published package will catch any exceptions and produce an output that indicates the error.

4.5 XSD and XSLT Lookup

As mentioned earlier, various pieces of information are taken from the input XML to decide how to create the output XML.

First the CALLER tag in the input document is examined and validated against a lookup table of valid subscribers for the service operation being called. If the lookup fails then the service call is aborted. If the lookup is successful then the information returned includes an (optional) XSLT document.

The input XML must conform to the XSD for that operation. When called, the published function looks up the schema document corresponding to the function and validates the passed in XML. This has the side effect of populating any default values in the XML as well as ensuring the correct grammar.

The output function is then called and the results are passed back into the published function. It should be noted that the back-end function does not add the enclosing OUTPUT tag. This tag is added in the published function so that the back-end function is not tied to a single web service operation.

At this point the XSLT document - if needed - can then be used by the published package function to transform the output XML produced by the back-end function.

The XSD for the output is now found and the result XML validated against it. Again, the validation process can fill in any defaults specified in the XSD.

Provided that the XML has passed XSD validation the results are returned back to the calling subscriber. If at any point an error has been raised then the published package will catch any exceptions and produce an output that indicates the error.

4.6 Third Party Integration

When a client wishes to subscribe to a web service operation they need to be given a pair of XSD documents. They specify the exact grammar required for the input and output XML documents.
If a client's needs are very similar to the vanilla service output, but they need the output in a different format, or need certain information stripping out of the output, then an XSLT must be created to transform the vanilla output and a new XSD must also be created.

The XSD allows both parties to be sure that the returned XML is understood and correct. The XSLT allows the vanilla service to be used to return a new flavour for the particular client.

XSLT transforms will only ever be applied to the output. If new input parameters are required or defaults for input parameters must be sought, then either a new published function must be written or the XSD for the input document must specify the default values.

5. BTe Store6 Interface

The BTe store6 system is highly configurable so the interface to Store6 has a number of variations. These are largely down to the way in which the integration gets designed in terms of mapping CIMS objects to Store6 and vice versa. There is also the predominant use of either the sys_barcode or the sty_barcode in CIMS to describe the SKU.

In addition to the variations that exist on the individual service operations it is also possible to "tweak" certain outputs using XSLT to alter the XML outputs and inputs.

5.1 Service Operations

The service operations are implemented in WebLogic application server and are driven by calls from BTe. Each operation expects a certain document to be given to it. The document is read and inserted into the relevant CDI tables for processing through to CIMS.
CIMS processing is deferred until the relevant i8x job is run to action the CDI table though into CIMS. Forms in CIMS exist to query queued / failed data.

importPosBanking

This operation expects to be given XML containing a POS or CASHOFF document. It uses the CDI pos_ftrans routines to import the data.
[PROLOGICCID-1225]

importPosTrans

This operation expects a POS document with IsSale or IsPaidIO set. Records are inserted into CIMS through the CDI pos_grp tables.
[PROLOGICCID-930, PROLOGICCID-931, PROLOGICCID-1223]

importRetailEmployee
This operation takes retail employee information for import into CIMS. It uses the CDI ret_emp tables for data import.
[PROLOGICCID-872]

importStockTransfer
A STOCK document is expected here and will be imported into CIMS using the CDI move interface.
[PROLOGICCID-1224]

5.2 Client Operations

The client operations are triggered by events in CIMS. They can be scheduled to run based on a queue or can be launched directly. The CIMS jobs are created using the i8x routines.
A Java component in the WebLogic server reads from a queue table to push XML documents out to a BTe server.

[PROLOGICCID-855]

fullItemDetailsUpdates
This operation sends to BTe a complete list of products, prices and barcodes. The registry table is first cleared and re-built so that a complete list is sent
.
Variations on this output centre on differences to Class and Department definitions and whether to include sys or sty barcodes.

Creates the InputItemData and InputStoreGroupItemPriceData documents.
[PROLOGICCID-1219]

itemDetailsUpdates
The registry table is queried to send only those items that have been updated or inserted since the last call.

Variations on this output centre on differences to Class and Department definitions and whether to include sys or sty barcodes.

Creates the InputItemData and InputStoreGroupItemPriceData documents.
[PROLOGICCID-1219]

itemDetailsDeletes
The registry table is queried for deletes since the last time the operation was called.

Variations on this output centre on differences to Class and Department definitions and whether to include sys or sty barcodes.

Creates the InputItemData and InputStoreGroupItemPriceData documents

[PROLOGICCID-1219]

retailEmployees
Export to store6 the retail employee data as held in CIMS.
Creates the
[PROLOGICCID-872]

storeDetails
This operation pushes details of stores up to the BTe service, including price list details for the store.
Creates the InputStoreGroupData document
[PROLOGICCID-1215]

stockItem
A registry table is used to return all the stock from the gar_stk table. If necessary, the output from this query can be sent to BTe in chunks - a complete stock list would (probably) be too large to send in a single document.

Creates the InputStockItemData document.
[PROLOGICCID-897, PROLOGICCID-898, PROLOGICCID-1220]

stockNotice
Return stock notices for items defined in CIMS for live products on the master shop price-list. The corresponding gtrans headers in CIMS are set to DONE.

Creates the InputStockNoticeData document.
[PROLOGICCID-899, PROLOGICCID-1222]

stockAdjust
Return stock adjustments for items defined in CIMS for live products on the master shop price-list. The corresponding gtrans headers in CIMS are marked as processed (dmt_state set to D).

Creates the InputStockNoticeData document.
[PROLOGICCID-1221]

storeDept
A document containing all the valid department details as mapped in CIMS is created with this endpoint.

Variations exist to map the store6 department with various entities in CIMS (i.e company division or a combination of retail type and product group).

Creates the InputDepartmentData document.
[PROLOGICCID-1216]

storeClass
This operation returns a document of classes defined in CIMS for live products on the master shop price-list, grouped by retail group.

Store6 class can be mapped to different entities in CIMS similar to the department mapping.

Creates the InputClassData document.
[PROLOGICCID-1217]

styles
Return styles defined in CIMS for live products on the master shop price-list, grouped by product hierarchy.

Creates the InputStyleData document.
[PROLOGICCID-1218]

6. CIMS Based Services

6.1 Feeds

Feeds use registry tables to keep track of entities in CIMS and allow changes in data to be sent out. The registry table is maintained by triggers applied to the relevant tables in CIMS.

If a 'full' request is made all the data in the registry is created as if nothing for that feed had ever been sent. The assumption is that the receiving end clears down any content as part of the process.
'Incremental' requests pull just the data that has been marked as changed from the registry tables.
It is possible to request feed data in chunks so that a given output will not result in a document that will take too long to either create or transfer.

Customer Feed
All CIMS customers are exported (loc_defs records where ploc_type = 'POS').
[PROLOGICCID-940]

Credit Note Feed
Credit, deposit and gift notes are exported from CIMS.

[PROLOGICCID-941]

Product list
All products that exist on the default retail price list will be included in the output for this service. The information includes a subset of data from STY_DEFS (PD0A) and the sku information as well. The main identifier for a product is the sys_barcode.

A variation of this feed exists with a simplified data set.
[PROLOGICCID-107, PROLOGICCID-108, PROLOGICCID-936]

Price list
Price list data for LIVE products that are web enabled are included in this output. Price lists are grouped by country and a new date against country price lists in CIMS has been added. Price lists will not be sent unless the send date has been reached.

A variation exists excluding the country data.
[PROLOGICCID-109, PROLOGICCID-110, PROLOGICCID-937]

Stock list
The stock list is generated from channel stock for the eCommerce channel and, due to the way that channel stock is calculated each night, the registry table operates in a different way to the standard registry tables which monitor changes. Instead, a complete picture of the output sent is maintained and a difference between the two sets is used as the basis for the next output.

Any stock in frozen bins is ignored and a percentage of the non-web channel stock can be included in the output.

A variation exists which limits output to pre-defined channel codes in pre-defined warehouses and omits the negative bin check.
[PROLOGICCID-111, PROLOGICCID-112, PROLOGICCID-938]

Transfer feed
Stock movements to and from a soft coded set of locations and stock types are exported.
[PROLOGICCID-939]

Order Status
This feed returns order status for the order list supplied. Status includes the associated dispatch state for each order item.
[PROLOGICCID-117]

6.2 Operations

Regular operations can provide real-time reporting of CIMS data or manipulate entities in CIMS via the CDI interfaces.

Order Creation
This operation takes in a complete order and will create entries in the CIMS customer database as required. Customer, billing and delivery addresses are identified by an externally maintained WEB_ID. The delivery address for an order can be a store, provided that store is marked as a buy & collect sore. The response for this service is the order status. It is not guaranteed that a despatch note can be created from the order; it is up to the calling process to ensure that there is sufficient stock before creating an order.

Orders are created using the CDI_SOR interface.
[PROLOGICCID-113, PROLOGICCID-114]

Order Amendment
Orders can be amended by this operation to apply a payment method, progress the order despatched into a state of SEND, or cancel a given order.

Orders are amended using the CDI_SOR_AMEND_PKG interface.
[PROLOGICCID-116, PROLOGICCID-145]

Order Returns
Returns notes (as visible in DI0K) can be created using this interface. In addition to returns to a warehouse it is also possible to return stock to stores. Note that there is no provision for exchanges.

Returns are created using the CDI_RET_PKG.
[PROLOGICCID-115, PROLOGICCID-118]

Store Stock Query
This operation provides real-time stock lookup for a given set of SKUs in a given location. The location can be specified as a store number (in which case the output is limited to just that store) or can be given as a set of co-ordinates or postal location in which case all stores are queried for the selected SKUs and returned in order of distance from the specified location.
[PROLOGICCID-144]

Pos Customers
This interface operation allows for the insert or update of CIMS PoS customers using the CDI_SOR import_customer routines. Externally identified customers are tagged using the WEB_ID field in CIMS (stored in loc_attributes).
[PROLOGICCID-950]

Pos Sales
Ingest PoS created sales, including creation of new customers (or update of existing).

[PROLOGICCID-944]

Pos Banking
Banking interface to the CDI pos_ftrans routines and tables.
[PROLOGICCID-946]

Pos Transfers
Creation of stock movements in CIMS.
[PROLOGICCID-948]

PoS Staff
Creation of staff records for tills.
[PROLOGICCID-943]

7. Upgrade Instructions

This release for Prologic Head Office CIMS can be obtained by contacting your Account Manager or PM to discuss pre-requisites and schedule an installation.