aptean

EBB Paper

Operational Documents

Functional Specification

21st May 2020 - 3.0
Reference: FS 371933 6287

Contents

1 Functional Description

1.1 Database & Data Access Layer

The following fields will be added to the Job table EPOD_REASON_CODE:

- EPL_CONV_TYPE - nvarchar(10)
- EPL_CONV_VALUE - float
- EPL_CONV_CODE - nvarchar(20)

These fields will be added to all stored procedures in the database that require them.

These fields are *not* required on the device.

These fields are *not* required to be set on codes imported through the standard XML import (flat file or webservice).

These fields are *not* required to be contained within the Export.

These fields are *not* required to be used when filtering data for selection.

The following fields will be added to the Job table EPOD_XF_CONFIG:

- EPL_DB_CONNECTION - nvarchar(255)

This field will be added to all stored procedures in the database that require it.

This field is *not* required on the device.

This field is *not* required to be used when filtering data for selection.

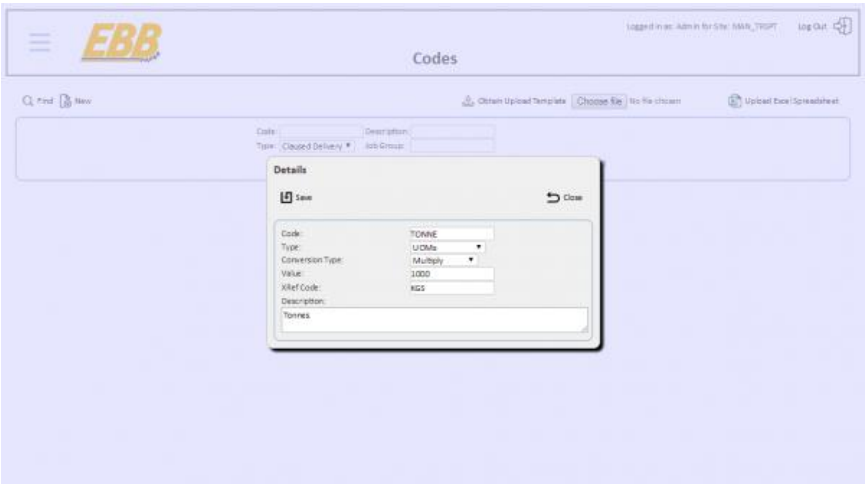
Note: All DAL objects that link to XF_CONFIG may also require change.

The EPOD_SETUP stored procedure will be modified for the new EPOD_LISTS and EPOD_LIST_ITEMS values.

1.2 Admin

1.2.1 Codes Maintenance Screen

The Codes Maintenance screen (reason_code.aspx) will be modified to support UOMs.

- 

Codes Maintenance

The **Find** criteria will be modified to allow selection of the new "UOM" type. Additionally, a blank "-- Select --" option will also be provided, for when the user wished to see all codes of all types. A new Code Type of "UOM" (description "UOMs") will be added to the EPOD_LIST_ITEMS for Reason Codes.

The results table does *not* require modifications.

The pop-up Entry/Edit form will be modified for entry of the new UOM codes.

This is accessed when entering a new code using the **New** button, and when editing existing codes by clicking the **Select** button against a line in the results table.

The user will be allowed to enter codes of type "UOM", selected from the drop-down list. A new Code Type of "UOM" will be added to the EPOD_LIST_ITEMS for Reason Codes.

When editing or adding reason codes of type "UOM" *only*, new fields will be present on the pop-up form, as shown below:

Label	Field	Pop-up Help
Conversion Type	EPL_CONV_TYPE	USED BY BESPOKE INTERFACES ONLY to determine whether the quantity is modified or the line processed when the UOM against the product line matches this value
Value	EPL_CONV_VALUE	USED BY BESPOKE INTERFACES ONLY. If Conversion Type is "Multiply", this value is used to multiply the product quantity. If Conversion Type is "Round", this is used to round the quantity to the nearest value provided here.
XRef Code	EPL_CONV_CODE	USED BY BESPOKE INTERFACES ONLY. If present, the UOM is converted to the value provided here.

These should be laid out as shown in the screenshot in the pop-up form.

The Conversion Type will be added with the following drop-down list items, populated from the EPOD_LISTS and EPOD_LIST_ITEMS tables.

- "Multiply" - enter numeric value for multiplication.
- "Round" - enter value to round to (e.g. 1 means integer value, 0.25 is round to nearest 0.25, etc)
- "Exclude" - lines of this DU type are not added at all - Value field disabled.
- "As Received" - no calculation - Value field disabled.

The default value for this list will be set by the default value in the EPOD_LIST_ITEMS table, as follows:

- Defaults to Type "Multiply" for EBB Paper
- Default to "As Received" for all other customers.

The Value field will be enabled only if the following values are selected in the Conversion Type:

- Multiply
- Round

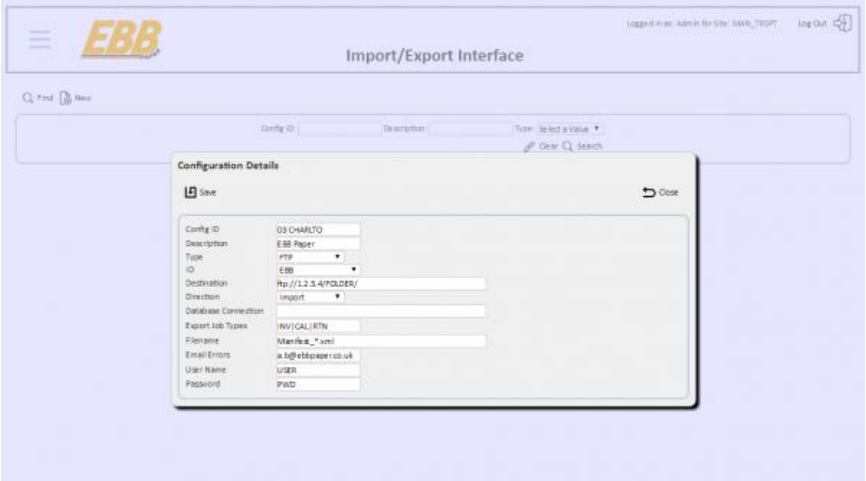
The Value field will be a numeric text field, allowing decimal entry, defaulting to "1000" if enabled.

The XRef Code will be a drop-down list of all codes of the Code Type being edited/added by this screen, and will be populated on selection of the Code Type. This drop-down list will include a blank default value of "Blank", and show the Code and Descriptions of all the selected codes. The list will be ordered by the Code, ascending.

When saving, the values will be saved in the appropriate fields on the EPOD_REASON_CODE table, as shown above.

1.2.2 Interface Config Screen

The Import/Export Maintenance screen (xf_config.aspx) will be modified to support configuration of the new EBB Paper interface.

- 

Import/Export Configuration Maintenance

Fields required to be displayed when configuring the "EBB" interface:

- EPL_XF_CONFIG_ID - Always displayed.
- EPL_DESCRIPTION - Always displayed.
- EPL_XF_TYPE - Always displayed.
- EPL_XF_DESTINATION - Always displayed.
- EPL_XF_ID - Always displayed. This drop-down list will be modified to include ID "EBB".
- EPL_XF_DIRECTION - Always displayed.
- EPL_WEB_USER - only if FTP or SOAP transfer types are selected.
- EPL_WEB_PASSWORD - only if FTP or SOAP type selected
- EPL_DB_CONNECTION - This new field will only display if type "EBB" is selected and will include validation that it must be entered. This will be labelled as "Database Connection". The field will be a wide entry field (at least double the width of a standard field).
- EPL_EXPORT_JOB_TYPES - Display if type is "EBB". The validation on this field will be modified so that pipe-delimited list of anything can be entered, if the type is "EBB".
- EPL_FILENAME - Filename to match when picking up files. Expected to be "Manifest_*.xml".
- EPL_EMAIL_ERRORS - email address for file processing error notifications.

1.3 Auto-Import

1.3.1 General

The process will update and create all data as required, committing each as they are processed.

There are instances where the failure reason will result in the following processing:

- Stop processing this file and all further files (Failed Import).
- Stop processing this file and process subsequent files (Failed File).
- Stop processing this order and move to the next order (Failed Order).

Failed Import reasons:

- Unable to connect to MSA

Failed File reasons:

- Standing Data not being created:
  - ♦ User does not exist
  - ♦ Vehicle does not exist
  - ♦ Job Group does not exist

- Non-DESK manifest is on a load that has already been completed

Failed Order reasons:

- No details of this order in MSA
- Job for this order already complete

Reasons why a Manifest file will not create Loads and Jobs:

- Site does not exist
- Manifest is a trunk.
- Jobs are of the wrong type.
- There are no valid products on the job.
- There are no valid jobs on a Load.

If there are issues processing, the whole file will be rejected and will need to be reprocessed. However, all data saved to that point will be present.

Audit records will be maintained, identifying successfully processed and failed import files, identifying the reason why a file failed to import.

It is possible for a manifest to generate multiple audit reasons when processing a file. All of these codes will be audited. Multiple audit records may be maintained per file, or a single audit record created with all details in it. This will be decided when developed, based on which method is a more generic solution for the C-ePOD product.

Rejected files will be emailed (depending on configuration) and will be stored in the standard Failed files location, as the processes do now.

Note: The process generating the email will be modified to include the auditing of the file.

When a file is reprocessed, this will over-write and update the existing data, as specified in the processing below.

The processing of each file will proceed as follows:

- Retrieve Manifest files.
- Filter and validate the manifest file.
- Create User and Vehicle data.
- Create a new Load or combine to an existing Load.
- Create or update jobs on the load, through a link to the MSA view.
- Create or update products on the jobs.
- Remove any products from the job not contained in the manifest file.
- Remove any jobs from the load not contained in the manifest file.

These areas are covered in the following sections

1.3.2 Retrieve Manifest Files

For FILE transfer type, currently all files in the specified destination folder (EPL_XF_DESTINATION) are processed (using Directory.GetFiles). This will be modified so that, if a filename pattern is specified (in EPOD_XF_CONFIG.EPL_FILENAME), the process will only retrieve and process files matching this pattern (by passing a second parameter to the GetFiles method, specifying the pattern from EPOD_XF_CONFIG.EPL_FILENAME).

For FTP transfer type, the same is true, in that all files are retrieved (using ListDirectory in method GetFileList). No facility exists to pattern-match file names. Therefore the method GetFileList will be modified so that, if a filename pattern is specified (to be passed to an overloaded method in a new parameter, passing EPOD_XF_CONFIG.EPL_FILENAME), the process will only add these to the result if the filename returned matches the pattern provided.

Regardless of whether the file was retrieved through FTP or from the filesystem, each file (representing a single manifest) will be passed to a new processing method to validate and process the contents and create jobs and products. The current process bases this on EPL_MSG_TYPE. This will be extended to check the EPL_XF_ID field - should this be "EBB", the new processing method will be called.

A file failing to process will not stop subsequent files from processing. However, if the MSA view cannot be connected to, processing will stop (see section Create Jobs for details on the MSA View connection).

1.3.3 Validate the Manifest

This process will receive the manifest contents as an XML file. This file will be downloaded before processing to ensure that all characters that should have been URL encoded have been. The file will be pre-parsed to remove any non printable characters.

The **depot** tag will be extracted from the **manifest/header** tag contents. The site will be retrieved using the EPL_EXT_FLEET field. If a site is not found with a fleet matching the **depot** tag, the manifest will not be processed.
Note: This is not an error and will not be audited as such - the manifest will be marked as successfully processed.

The site retrieved will be stored by the process in an EPOD_SITE DAL object.

The **trailer** tag will be extracted from **manifest/header** tag contents. If this tag starts with the text "TRUNK", then this manifest will not be processed.

1.3.4 Create Vehicles and Users

The **driver** tag will be extracted from the **manifest/header** tag contents.

If this driver is set to "WAREHOUSE", the user ID will be created as this value.

If this is a non-zero value that is not explicitly "WAREHOUSE", C-ePOD will attempt to create the driver ID (if configured to do so) as the first character of the first name, and up to the first 4 characters of the surname, with a 3-digit unique identifier added to them. The case will be converted to uppercase.

For example:

- "David Eastman" will become "DEAST001"
- "SMITH JOHN" will become "SJOHN001"
- "DEREK MAY" will become "DMAY001"
- "TRUNKER M25". TRUNKER M25 will become "TM25001"

The process will attempt to find this user using the Site ID and the generated User ID above. If a record is found, the process will compare the user name in the XML file to the user name on the database. If different, the 3-digit id will be incremented by 1 and the process repeated until a matching EPOD_USER record is found, or one is not found for this ID.

Check the EPOD_SITE.EPL_IMPORT_CREATE_SD_FLAG. If this is set to "N" and an EPOD_USER record is not found, the file should be rejected. An audit record of "User X Not Found" shall be created.

If one is not found and EPOD_SITE.EPL_IMPORT_CREATE_SD_FLAG is "Y", the EPOD_USER record will be created and updated as follows:

Field	Populated by
EPL_SITE_ID	EPOD_SITE.EPL_SITE_ID
EPL_USER_ID	either "WAREHOUSE", or as generated above
EPL_USER_PASSWORD	set to the same value as the user ID
EPL_USER_NAME	**driver**
EPL_USER_ADMIN	"N"
EPL_USER_ACTIVE	"Y"

Field	Populated by
EPL_USER_ACTIVITY_DATE	0
EPL_USER_ACTIVITY_TIME	0
EPL_PASSWORD_VISIBLE_IND	1

The **vehiclereg** tag will be extracted from the **manifest/header** tag contents. If this is a non-zero value, C-ePOD will attempt to create the driver ID (if configured to do so).

The process will attempt to find this vehicle using the Site ID and the Vehicle ID as the Vehicle Reg above. If a record is found, this will be stored by the process as EPOD_VEHICLE.

Check the EPOD_SITE.EPL_IMPORT_CREATE_SD_FLAG. If this is set to "N" and an EPOD_VEHICLE record is not found, the file should be rejected. An audit record of "Vehicle X Not Found" shall be created.

If one is not found and EPOD_SITE.EPL_IMPORT_CREATE_SD_FLAG is "Y", the EPOD_VEHICLE record will be created and updated as follows:

Field	Populated by
EPL_SITE_ID	EPOD_SITE.EPL_SITE_ID
EPL_VEHICLE_ID	**vehiclereg**
EPL_VEHICLE_REG	**vehiclereg**
EPL_DESCRIPTION	**vehiclereg**
EPL_STATUS	"Y"

Note: Data created on these tables will be truncated. Comparisons to this data as described above with be truncated before comparison, in case the data is space-filled in the XML file.

1.3.5 Create a Load

A load will be attempted to be found using the following data, providing that at least a vehicle or user has been provided:

- EPL_SITE_ID - EPOD_SITE.EPL_SITE_ID
- EPL_LOAD_START_PLANNED_DATE - **deliverydate** tag. Note that this value must be reformatted to YYYYMMDD format from DD-MM-YYYY format.
- EPL_VEHICLE_ID - EPOD_VEHICLE.EPL_VEHICLE_ID if one is present, otherwise this is not set as an parameter for the selection.
- EPL_USER_ID - EPOD_USER.EPOD_USER_ID if one is present, otherwise this is not set as an parameter for the selection.
- EPL_STATUS - Not equal to "C" "X" if the **driver** attribute is not explicitly set to "WAREHOUSE", otherwise this is not set as an parameter for the selection.

If one is not found, a new load will be created on EPOD_LOAD as follows:

Field	Populated by
EPL_SITE_ID	EPOD_SITE.EPL_SITE_ID
EPL_LOAD_ID	generated by the EPOD system.
EPL_LOAD_START_PLANNED_DATE	extracted from the **deliverydate** tag. Note that this value must be reformatted to YYYYMMDD format from DD-MM-YYYY format.
EPL_LOAD_START_PLANNED_TIME	extracted from the **starttime** tag. Note that this value must be reformatted (removing the embedded colon character and adding "0000" to the end. Note that if this tag is not present or blank, this field should be set to 6000000.
EPL_LOAD_END_PLANNED_DATE	as the Load Start Planned Date.
EPL_LOAD_END_PLANNED_TIME	0
EPL_VEHICLE_ID	EPOD_VEHICLE.EPL_VEHICLE_ID
EPL_USER_ID	EPOD_VEHICLE.EPL_USER_ID
EPL_STATUS	"P"
EPL_LOAD_INFORMATION	the *number* attribute of the **manifest** tag concatenated with the **addinstruct** tag, delimited by a dash.

If a load is found and the manifest is *not* for desk collection (i.e. where the **driver** tag value is not explicitly

"WAREHOUSE"), the status (EPL_STATUS) should be checked. If this value is not "P" then the file should be rejected. An audit record of "Load X at wrong status" shall be created.

If a load is found, the Load Information field EPL_LOAD_INFORMATION should be checked that it contains the text built from:

- the *number* attribute of the **manifest** tag concatenated with the **addinstruct** tag, delimited by a dash.

If it does not contain this, this should be added to the text already in the field, delimited by a newline character.

The load (found or created) should be stored in an EPOD_LOAD object and updated.

1.3.6 Create Jobs

The process will find and store the highest sequence of any jobs already attached to the EPOD_LOAD object, by checking the content of the EPOD_JOBS list on the EPOD_LOAD object.

The process will extract the list of allowed order prefixes from the EPOD_XF_CONFIG.EPL_EXPORT_JOB_TYPES field.
Note: If this parameter is empty, all orders will be processed.

The process will loop through all instances of the **order** tag in the *manifest/data* tag. Note that this is separated by a **customer** tag.

Each **order** tag will be processed to create EPOD_JOB records, which will be added to the EPOD_LOAD.EPOD_JOBS list.

The *sopnumber* attribute of the **order** tag will be extracted. if the attribute does not exist or has no value, this tag may be skipped.

The trimmed attribute value will be checked to see whether the value begins with any of the allowed order prefixes. If it does not, then then the order will not be processed.

A connection will be made to the MSA database for the order details. This will be achieved by opening a connection using the connection information stored in EPOD_XF_CONFIG.EPL_DB_CONNECTION. The connection should be attempted several times (for example, 3 times). If a connection cannot be made, then the file should be rejected. An audit record of "Order X: Cannot connect to MSA" shall be created. **Note:** As this is a fundamental requirement of the process, should this fail, processing of this file *and all others* should be terminated for this run.

The connection details will be specified in a connection string in EPOD_XF_CONFIG.EPL_DB_CONNECTION.

Note: At this time, the following information has not yet been confirmed.

- Connection details (IP, user, password, port)
- Database type
- View name

Note: This MSA view is used from this point onwards for all details. Should the connection fail when attempting to retrieve subsequent records from the MSA view, the connection should be attempted to be re-established, again retrying several times, to ensure that this process is predominantly uninterrupted.

In the instance where the connection fails, the file need not be rejected and moved to the Failure area - it can be left to be reprocessed immediately on the next run.

When the connection is established, the details of the order will be retrieved from the MSA view using a standard SQL query, querying on the SOP Number.

Should the view return no details, then the order will not be processed. An audit record of "Order X: No details in MSA" shall be created. The process will continue with the next order in the XML file.

The Load will be checked to see whether the order already exists - the process will look for the order in EPOD_LOAD.EPOD_JOBS, where the job code (EPL_JOB_CODE_ is equal to the trimmed value of the **sopnumber** tag. If an order is found, an EPOD_JOB object will be created and set to the value of the job in the

EPOD_LOAD.EPOD_JOBS EPOD_JOB object by reference.

If the job found is already complete (EPL_STATUS = "C"), then the process will skip updating this order. An audit record of "Order X: already complete" shall be created. The process will continue with the next order in the XML file.

If a job is not found, a new job will be created using the key values:

- EPL_SITE_ID - EPOD_LOAD.EPL_SITE_ID.
- EPL_LOAD_ID - EPOD_LOAD.EPL_LOAD_ID.
- EPL_JOB_TYPE - If the left 3 characters of the **sopnumber** tag are "RTN", set this value to "C", otherwise set this value to "D".
- EPL_JOB_CODE - the trimmed value of the **sopnumber** tag.

A value in EPL_JOB_ID will be generated for this job at this stage.

The Job Group should be calculated as follows:

- If the EPL_JOB_TYPE is "C", set to "COL"
- else if EPOD_LOAD.EPL_USER_ID = "WAREHOUSE", set to "DESK"
- else set to "DEL"

The process will attempt to find this Job Group using:

- EPL_SITE_ID - EPOD_LOAD.EPL_SITE_ID.
- EPL_JOB_GROUP - as set above.

If a record is found, this will be stored by the process as an EPOD_JOB_GROUPS DAL object.

Check the EPOD_SITE.EPL_IMPORT_CREATE_SD_FLAG. If this is set to "N" and an EPOD_JOB_GROUPS record is not found, the file should be rejected. An audit record of "Job Group X Not Found" shall be created.

If one is not found and EPOD_SITE.EPL_IMPORT_CREATE_SD_FLAG is "Y", the EPOD_JOB_GROUPS record will be created as follows:

- EPL_SITE_ID - EPOD_SITE.EPL_SITE_ID
- EPL_JOB_GROUP - as set above.
- EPL_DESCRIPTION:
    - If EPL_JOB_GROUP is "COL", set to "Collections".
    - If EPL_JOB_GROUP is "DEL", set to "Deliveries".
    - If EPL_JOB_GROUP is "DESK", set to "Desk Collections".

All flags will be set as their default values.

The customer code, invoice address and contact information will be retrieved from the MSA view. The customer code will be checked as to whether it is already created in the system, by checking for an EPOD_CUSTOMER record using the keys:

- EPL_SITE_ID - EPOD_SITE.EPL_SITE_ID
- EPL_CUSTOMER_CODE - column CUSTNMBR from the MSA view.

If this record does not exist, or a record does exist but the details are different, the record will be updated as follows: If one is not found, a new customer will be created on EPOD_CUSTOMER as follows:

Field	Populated by
EPL_SITE_ID	EPOD_SITE.EPL_SITE_ID
EPL_CUSTOMER_CODE	CUSTNMBR
EPL_CUSTOMER_NAME	CUSTNAME
EPL_ADDRESS_1	InvAdd1
EPL_ADDRESS_2	InvAdd2
EPL_ADDRESS_3	InvAdd3
EPL_ADDRESS_4	InvCity

Field	Populated by
EPL_ADDRESS_5	InvState
EPL_POSTCODE	InvZip (Spaces removed and truncated to 8 characters)

Note: All values above are leading/trailing space trimmed unless otherwise stated.

The delivery address and contact information will then be compared to this Invoice address, comparing the following items:

- EPL_CUSTOMER_NAME - DelName
- EPL_ADDRESS_1 - DelAdd1
- EPL_ADDRESS_2 - DelAdd2
- EPL_ADDRESS_3 - DelAdd3
- EPL_ADDRESS_4 - DelCity
- EPL_ADDRESS_5 - DelState
- EPL_POSTCODE - DelZip

If any of the values are different, a Job Address will be created through an EPOD_JOB_ADDRESS DAL object, populated as follows:

Field	Populated by
EPL_SITE_ID	EPOD_SITE.EPL_SITE_ID
EPL_JOB_ID	EPOD_JOB.EPL_JOB_ID
EPL_JOB_TYPE	EPOD_JOB.EPL_JOB_TYPE
EPL_NAME	DelName
EPL_ADDRESS_1	DelAdd1
EPL_ADDRESS_2	DelAdd2
EPL_ADDRESS_3	DelAdd3
EPL_ADDRESS_4	DelCity
EPL_ADDRESS_5	DelState
EPL_POSTCODE	DelZip (Spaces removed and truncated to 8 characters)

Note: All values above are leading/trailing space trimmed unless otherwise stated.

If this is not the first job being processed for the load, the process will check whether the job being created is not the first job being created for the customer code in this manifest. If this is not a subsequent job for this customer code, blank any stored values for the following fields:

- EPL_SEQUENCE
- EPL_LINKED_ID

If this is not a subsequent job for this customer code, the process will then loop through existing jobs in EPOD_LOAD.EPOD_JOBS in reverse sequence, looking for any job that matches as follows:

- EPL_CUSTOMER_CODE is the same
- EPL_STATUS not "C" or "X"
- EPL_JOB_TYPE is the same as the job being created.
- The delivery address is the same as the job being created.

If a match is found, check the value of the field EPL_LINKED_ID. If there is no value, set this to the value in the record's EPL_SEQUENCE field and update the object. The process will then store the values in the following fields:

- EPL_SEQUENCE - EPL_SEQUENCE of the job found
- EPL_LINKED_ID - EPL_LINKED_ID of the job found

The EPOD_JOB object will then be updated with the values as follows:

Field	Populated by
EPL_JOB_GROUP	The calculated Job Group above
EPL_JOB_INSTRUCTION	Manifest No (manfest), Order comments (OrdComm) and Delivery Instructions (DelIns) concatenated together with a space separator.
EPL_STATUS	"P"

Field	Populated by
EPL_CUSTOMER_CODE	The customer code (CUSTNMBR).
EPL_SEQUENCE	Stored value of EPL_SEQUENCE if there is one, or the maximum value of sequences of jobs on this load, plus 1.
EPL_START_PLANNED_DATE	ReqShipDate
EPL_START_PLANNED_TIME	StartTime if populated, else 06000000 (06:00)
EPL_END_PLANNED_DATE	ReqShipDate
EPL_END_PLANNED_TIME	DelTime if populated, else FinishTime if populated, else 14000000 (14:00).
EPL_JOB_CODE	the trimmed value of the **sopnumber** tag.
EPL_CUST_REF	
EPL_OFFICE_INSTRUCTION	tcsFLST_Route
EPL_SO_NUMBER	
EPL_ORDER_DATE	CREATDDT
EPL_SALES_CONTACT	UserID
EPL_OWNER_NAME	The sales territory (SALSTERR)
EPL_EXT_REF	The manifest (extracted from the *number* attribute of the **manifest** tag)
EPL_ACCOUNT	
EPL_LINKED_ID	Stored value of EPL_LINKED_ID if there is one.
EPL_TIMEZONE	
EPL_LOAD_LOCATION	Originating Depot
EPL_TTM_STOP_NUMBER	Set to the value in EPL_SEQUENCE

Note: that if updating a job at EPL_STATUS of "X", any existing products/containers associated with this job will also be reset to pending status.

The process will then store the following:

- EPL_CUSTOMER_CODE - the Customer code last processed in this manifest file.
- EPL_SEQUENCE - EPL_SEQUENCE of the job created
- EPL_LINKED_ID - EPL_LINKED_ID of the job created

The job will be updated and saved at this point.

The Job ID of the job created will be added to a comma-delimited string of jobs processed for this manifest.

1.3.7 Create Products

The process will loop through each record from MSA view, in order to create the product lines. **Note:** All lines in the MSA view detail product information, including the first line.

The value in UOFM field of the MSA view will be looked up against the EPOD_REASON_CODE table matching the following parameters:

- EPL_SITE_ID - EPOD_JOB.EPL_SITE_ID
- EPL_REASON_CODE - UOFM

If a record is not found, create an EPOD_PRODUCT DAL object with values set as follows:

- EPL_SITE_ID - EPOD_JOB.EPL_SITE_ID
- EPL_REASON_CODE - UOFM
- EPL_DESCRIPTION - UOFM
- EPL_CONV_TYPE - "MULTIPLY"
- EPL_CONV_VALUE - 1000
- EPL_CONV_CODE - ""

Update this UOM code.

If the value of EPOD_PRODUCT.EPL_CONV_TYPE is "EXCLUDE", this product will not be added - the process will move to the next record in the MSA view.

If the value of EPOD_PRODUCT.EPL_CONV_TYPE is "MULTIPLY", multiply the value in the QUANTITY field of the MSA view by the value stored in EPL_CONV_VALUE and store as the quantity.

If the value of EPOD_PRODUCT.EPL_CONV_TYPE is "ROUND", round the value in the QUANTITY field of the MSA view by the value stored in EPL_CONV_VALUE and store as the quantity.

If the value of EPOD_PRODUCT.EPL_CONV_TYPE is "DEFAULT", store the value in the QUANTITY field of the MSA view as the quantity.

If the value of EPOD_PRODUCT.EPL_CONV_CODE has a valid value, store the EPL_CONV_CODE as the UOM.

If the value of EPOD_PRODUCT.EPL_CONV_CODE is blank or null, store the UOFM field from the MSA view as the UOM.

The process will check to see if a product already exists on the job by checking the EPOD_JOB.EPOD_PRODUCTS list for an object with the following values:

- EPL_SITE_ID - EPOD_JOB.EPL_SITE_ID
- EPL_JOB_ID - EPOD_JOB.EPL_JOB_ID
- EPL_CONTAINER_ID - "000000000000000" (15 zeroes in a string, denoting a loose product)
- EPL_PRODUCT_CODE - ITEMNMBR
- EPL_SEQUENCE - linenumber/16384 (first 4 characters of this if not a whole number)

Note that ITEMNMBR and ITEMDESC may contain control characters not compatible with the EPOD device software. These should be stripped out before saving into the EPOD database.

If one is not found, an EPOD_PRODUCT object will be created and added to the EPOD_JOB.EPOD_PRODUCTS list.

If one is found or created above, an EPOD_PRODUCT object will be pointed to this object by reference.

The EPOD_PRODUCT object (found or created) will have the fields updated as follows:

Field	Populated by
EPL_SITE_ID	EPOD_JOB.EPL_SITE_ID
EPL_JOB_ID	EPOD_JOB.EPL_JOB_ID
EPL_CONTAINER_ID	"000000000000000"
EPL_PRODUCT_CODE	ITEMNMBR
EPL_SEQUENCE	linenumber/16384 (first 4 characters of this if not a whole number)
EPL_DESCRIPTION	ITEMDESC, truncated to 40 characters
EPL_PRODUCT_QTY_PLANNED	The quantity calculated and stored above
EPL_PRODUCT_QTY_CASE	SellPack if populated and numeric, else 0.
EPL_STATUS	"P"
EPL_PRODUCT_WEIGHT	ITEMSHWT
EPL_CUST_REF	CustordNo
EPL_ITEM_TYPE	
EPL_UNIT_TYPE	The UOM stored above.
EPL_DESCRIPTION_LONG	ITEMDESC and ebbFLTX_Sales_Comments concatenated with CR/LF characters.
EPL_PRODUCT_QTY_ORDERED	Set to the same value as EPL_PRODUCT_QTY_PLANNED

The product record on EPOD_PRODUCT will then be updated.

The Product Code of the product created will be added to a comma-delimited string of products processed for this job.

1.3.8 Remove Products not on the Job

All subsequent records on the MSA view will be processed in this manner and stored on the job.

When all products are processed, the process will check through all products on the job and compare to the list of products processed on this manifest for this job. Any products on the job but not in the manifest will be deleted.

If, after this, there are no products on the job, the job will be deleted. An audit record of "No Products on Order X - order deleted" shall be created.

1.3.9 Remove Jobs not on the Load for that Manifest

All subsequent order tags in the Manifest will be processed in this manner and stored on the load.

When all orders on the manifest are processed, the process will check through all jobs on the load (with that Manifest number, as stored in EPL_EXT_REF) and compare to the list of jobs processed on this manifest. Any jobs for this manifest on the load but not in the manifest will be deleted.

If, after this, there are no jobs on the load, the load will be deleted. An audit record of "No Jobs on Load X - load deleted" shall be created.

2 Appendix A: TEST PLAN

Test Script / Scenario Reference	*Bespoke Import Interface*	**Call Number(s): 344273 SCR-343463-01**
Test Script / Scenario Description	*Testing the new EBB paper interface can be configured and works as expected.*	**PASS / ISSUES / FAIL**
Menu Access	*N/A*	
Pre-requisites	*A system configured as EBB Paper.*	**Tested By:**
Test Objective	*To test that: UOMs may be entered with conversion types; the EBB Paper interface may be configured and; the EBB paper interface works as expected.*	**Date:**

Step	Action	Result	Remarks	P/F
1	**Admin**			
1.01	Create a new Interface type as expected for EBB Paper.	The new EBB Paper interface can be configured and edited as expected.		
1.02	Create a new reason code. Select type "UOM".	Type UOM can be selected from the drop-down list. The Conversion fields are displayed and defaulted as expected.		
1.03	Select Conversion Type "Round" and "Multiply".	A Value may be entered.		
1.04	Save the code. Enter another code. Select other Conversion Types.	A Value may not be entered.		
1.05	Save the code. Enter another code with an Xref code and conversion type "As Received".	The XRef code is entered and saved.		
1.06	Find all UOM codes.	UOM can be selected as a filter element. All UOM codes are displayed.		
1.07	Edit UOM codes.	The codes are displayed as they were entered. The Conversion fields are displayed as expected.		

Step	Action	Result	Remarks	P/F
2	**Interface**			
	Set up the site to NOT create standing data. Configure the interface for FTP file transfer. Configure the MSA connection to be incorrect.			

2.01	Create valid manifest files in the configured FTP area, one named "Manifest_test1.xml" and one named "incorrect.xml". Both should have a user that does not match an existing vehicle. Run the import process.	Only the manifest matching the filename is processed. This is rejected for an invalid user. The file is placed in a Failed directory.		
2.02	Change the interface configuration to FILE transfer type. Create valid manifest files in the configured FILE area, one named "Manifest_test2.xml" and one named "incorrect.xml". Both should have a vehicle that does not match an existing vehicle, but with a valid user. Run the import process.	Only the manifest matching the filename is processed. This is rejected for an invalid vehicle. The file is placed in a Failed directory.		
2.03	Process a manifest (any transfer method) with a valid user and vehicle, with job groups not created.	The file is rejected for invalid job group. The file is placed in a Failed directory.		
2.04	Configure the system to create standing data. Create two valid (but different) manifests, with a user and vehicle that does not match any existing data. Ensure there is a valid job with at least one valid product.	The first file is processed and rejected due to being unable to connect to MSA. The second file is not processed at all.		
2.05	Configure the interface with the correct MSA view connection details. Process the manifests above again.	The files are processed without errors. The Vehicle is created. The Users is created. The Job Group is created. The Customer is created. The Load is created. The Job is created. The products are created. The file is audited as processed successfully. The file is placed in a Success directory.		
2.06	Create a manifest almost identical to the last manifest, but with different Load start time, Job Start time, product quantities. Ensure the invoice address is changed. Ensure that there are comments on the Manifest. Process the manifest.	The load, job and products are updated. The manifest comments are appended to the Load Instructions. The invoice (customer) address is updated.		
2.07	Create a manifest for the same date, vehicle and user, with orders with one valid sopnumber, and all others as invalid SOP numbers. Ensure the valid order is not for the same customer as the prior manifest. Process the manifest.	A job is created on the load for the one valid order. All others are not processed. The job is sequenced after all others		
2.08	Create a manifest for the same date, vehicle and user, with an order with a valid INV sopnumber. Ensure the order is for the same customer as the first manifest. Process the manifest.	A job is created on the load for the order. The job is sequenced and linked to the matching job from the first manifest.		
2.09	Create a manifest for the same date, vehicle and user, with orders with a valid sopnumber with the same customer code as the last manifest. Ensure two are RTN jobs and two are INV jobs. Process the manifest.	Jobs are created on the load for the orders. The jobs are correctly identified as Collection and Delivery jobs, with the correct Job Group. The jobs are sequenced after any existing jobs on the load. The RTN jobs will have the same sequence and linked ID. The INV jobs will have the same sequence and linked ID as the		

		existing INV job on the load for this customer.		
2.10	Create a manifest for the same date, vehicle and user, with orders with a valid sopnumber with the same customer code as the last manifest. Ensure two are RTN jobs and two are INV jobs. Ensure the delivery address is different to the invoice address. Process the manifest.	Jobs are created on the load for the orders. The jobs are correctly identified as Collection and Delivery jobs, with the correct Job Group. The jobs are sequenced after any existing jobs on the load. The RTN jobs will have the same sequence and linked ID. The INV jobs will have the same sequence and linked ID. No jobs will be linked to existing jobs on the manifest. The Job Address is created for these loads.		
2.11	Create a manifest for a different date, vehicle and user, with orders with valid sopnumbers, for many customers. Ensure that they have a mix of UOMs configured with Round, Multiply, As Received, Exclude and Xref Code conversions, as well as a product without a configured UOM.	The jobs and products are created as expected. The Exclude product is not created. The quantities are set as per the conversion type rules. The unknown UOM is created with default values.		
2.12	Complete or cancel the load created. Process the same manifest file.	The manifest is rejected for invalid load status.		
2.13	Process a DESK COLLECT manifest with a user of "WAREHOUSE"	A load is created. The user is created.		
2.14	Complete or cancel the load created. Process another DESK COLLECT manifest file.	A new load is created.		
2.15	Process a manifest with a job with no valid products.	The file is processed successfully. Audit history is recorded as the job having no products, and the load having no jobs.		

3 Appendix B: Quote & Document References

Cost Details				
Activity	**Estimate No. of Days**	**No. of Days**	**Rate per Day (?)**	**Cost (? Exc. VAT)**
Requirements	0.00	0.00	750	?0.00
Change Request Evaluation	0.00	0.00	750	?0.00
Functional Specification	2.50	0.00	750	?0.00
Technical Specification	0.00	0.00	750	?0.00
Development	15.50	5.00	750	?3,750.00
Testing and Release	3.50	0.00	750	?0.00
Implementation	1.25	0.00	750	?0.00
Project Management	1.50	0.00	750	?0.00
TOTAL	24.25	5.00		?3,750.00

Estimate excludes training, release to live and go live support.

B.1 References

Ref No	Document Title & ID	Version	Date
1	REQ 343463 EBB Paper Solution Design	2.0	01/08/2017

B.2 Glossary

Term	Definition
EPOD	Electronic Proof of Delivery. The OBS EPOD system is *CALIDUS* ePOD.
CALIDUS eSERV	The OBS mobile system to complete Service functionality in the field. This is part of the *CALIDUS* ePOD system.
PDA	The mobile device on which the C-ePOD system will run in the field. This can be a Phone, EDA or industrial PDA, running Android.
DAL	Data Access Layer. A mechanism for accessing data by the system that is removed from the application, allowing for simplified access and providing protection to the data, as only approved DAL methods can be used to modify it.
GPS	Global Positioning System. A mechanism of retrieving accurate positioning information in the form of Latitude and Longitude (Lat-Long) co-ordinates from a device.
GPRS, 3G, HSDPA, Data Service	All terms referring to mobile device network connectivity, and the speed at which the device connects to the internet.

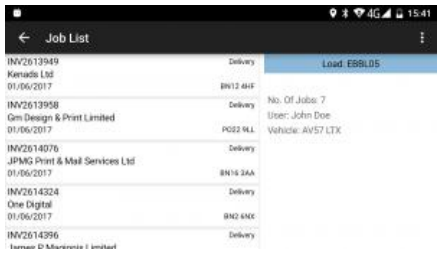
B.3 Authorised By

Matt Turner	OBSL Account Manager	_____
Rebecca Elliott	EBB Paper Representative	_____

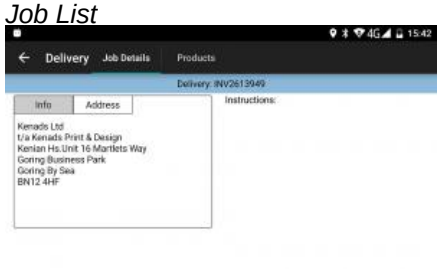
Login

- 

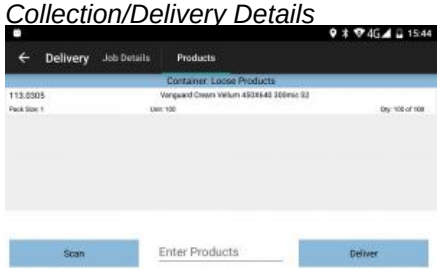
Numeric password

- 

Job List

- 

Collection/Delivery Details

- 

Products List

- 

Products Info

- 

Signature

4 Appendix A: TEST PLAN

Test Script / Scenario Reference	*Custom Mobile Device Styling*	**Call Number(s): 344275 SCR-343463-03**
Test Script / Scenario Description	*Testing the new EBB Paper mobile device style*	**PASS / ISSUES / FAIL**
Menu Access	*N/A*	
Pre-requisites	*A system configured as EBB Paper.*	**Tested By:**
Test Objective	*To test the new EBB Paper style.*	**Date:**

Step	Action	Result	Remarks	P/F
1	**Mobile Device**			
	Ensure there are "EBB Paper"-type jobs available for execution on the device.			
1.01	Configure to the "EBB Paper" style. Load the mobile device app on a device and complete a job.	The login screen should show the correct logo. The Job list should be configured as expected. The "Deliver Checked" check box should not be present. The Products list should be as expected. No Signatory should be present on customer signature.		

5 Appendix B: Quote & Document References

Cost Details				
Activity	**Estimate No. of Days**	**No. of Days**	**Rate per Day (?)**	**Cost (? Exc. VAT)**
Requirements	0.00	0.00	750	?0.00
Change Request Evaluation	0.00	0.00	750	?0.00
Functional Specification	0.25	0.00	750	?0.00
Technical Specification	0.00	0.00	750	?0.00
Development	0.25	0.00	750	?0.00
Testing and Release	0.25	0.00	750	?0.00
Implementation	0.25	0.00	750	?0.00
Project Management	0.00	0.00	750	?0.00
TOTAL	1.00	0.00		?0.00

Estimate excludes training, release to live and go live support.

B.1 References

Ref No	Document Title & ID	Version	Date
1	REQ 343463 EBB Paper Solution Design	2.0	01/08/2017

B.2 Glossary

Term	Definition
EPOD	Electronic Proof of Delivery. The OBS EPOD system is *CALIDUS* ePOD.
CALIDUS eSERV	The OBS mobile system to complete Service functionality in the field. This is part of the *CALIDUS* ePOD system.
PDA	The mobile device on which the C-ePOD system will run in the field. This can be a Phone, EDA or industrial PDA, running Android.
DAL	Data Access Layer. A mechanism for accessing data by the system that is removed from the application, allowing for simplified access and providing protection to the data, as only approved DAL methods can be used to modify it.
GPS	Global Positioning System. A mechanism of retrieving accurate positioning information in the form of Latitude and Longitude (Lat-Long) co-ordinates from a device.
GPRS, 3G, HSDPA, Data Service	All terms referring to mobile device network connectivity, and the speed at which the device connects to the internet.

B.3 Authorised By

Matt Turner OBSL Account Manager _____

Rebecca Eliott Customer Representative _____