Author	**Steve Baker**
Creation Date	**26th September 2014**
Created For	**General**

iMetal Design Document

FS-IM0010

Stock Allocation Import

Change Control

Issue	Date	By	QC	Issue	Summary
1	26/09/2014	SB			First Release
2	27/04/2018	SB			Add automated process type field

Table of Contents

1. Introduction

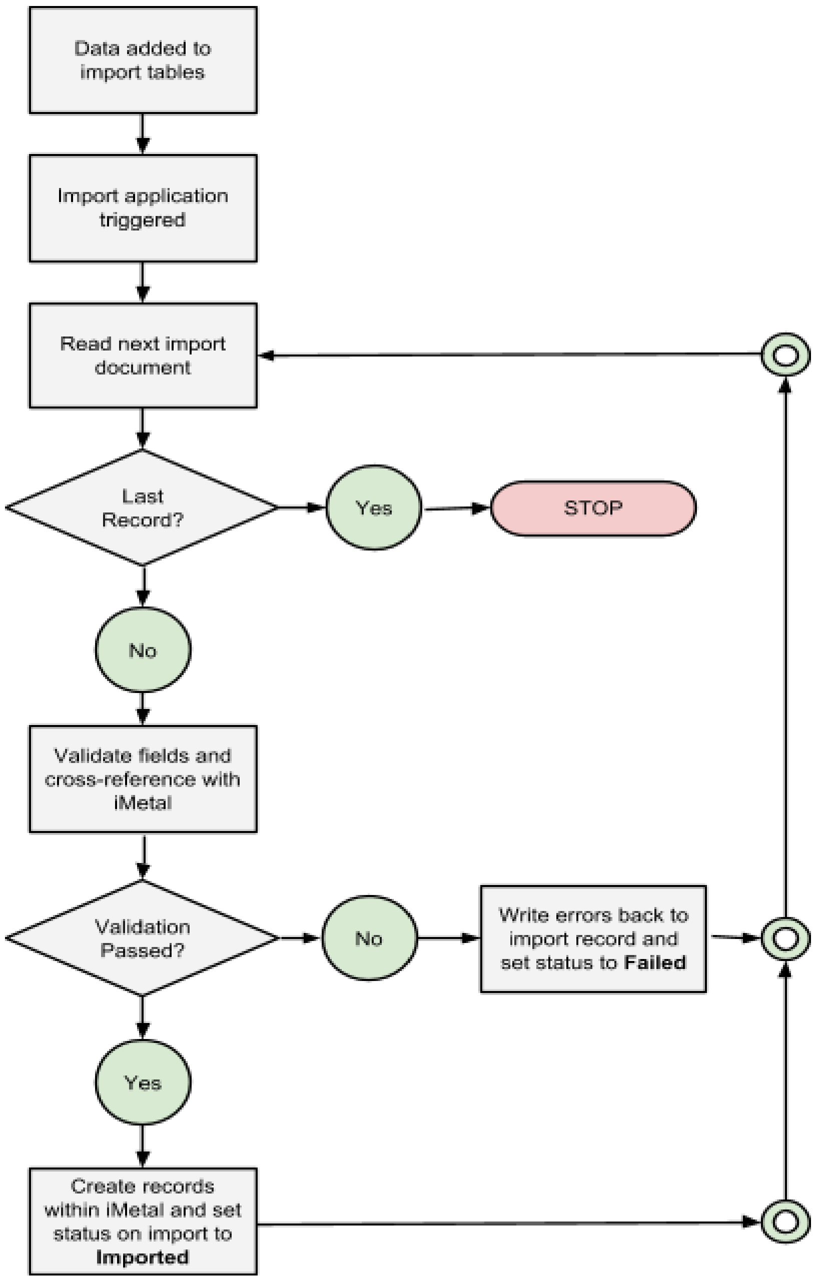
This document describes a generic Stock Allocation Import application that will provide the ability to import stock allocations into iMetal from external sources. The import will provide the following features:

- Simple import table structure within SQL database
- Fully de-referenced data fields (i.e. no ID's used)
- Validation of all import fields
- Common import database can feed multiple iMetal systems
- Screen-based and Command-line import options
- The initial release will be for stock allocations against Sales Items
- Ability to delete stock allocations

2. Core Flow

Although the stock allocation import could be used in various different scenarios, the common core functionality is as follows:

1. New stock allocation data is added to the stock allocation import table
2. The iMetal import application is triggered
3. Each stock allocation in the import data is validated, cross-referencing it with iMetal tables
4. Any errors during the validation are written back to the import record(s) and the import of that allocation is abandoned. An *import status* field on the import allocation is updated as 'Failed'
5. If the validations pass, then a new allocation is created within iMetal and the import status on the header is set to 'Imported'


Data added to import tables
Import application triggered
Read next import document
Last Record?
Yes
STOP
No
Validate fields and cross-reference with iMetal
Validation Passed?
No
Write errors back to import record and set status to **Failed**
Yes
Create records within iMetal and set status on import to **Imported**

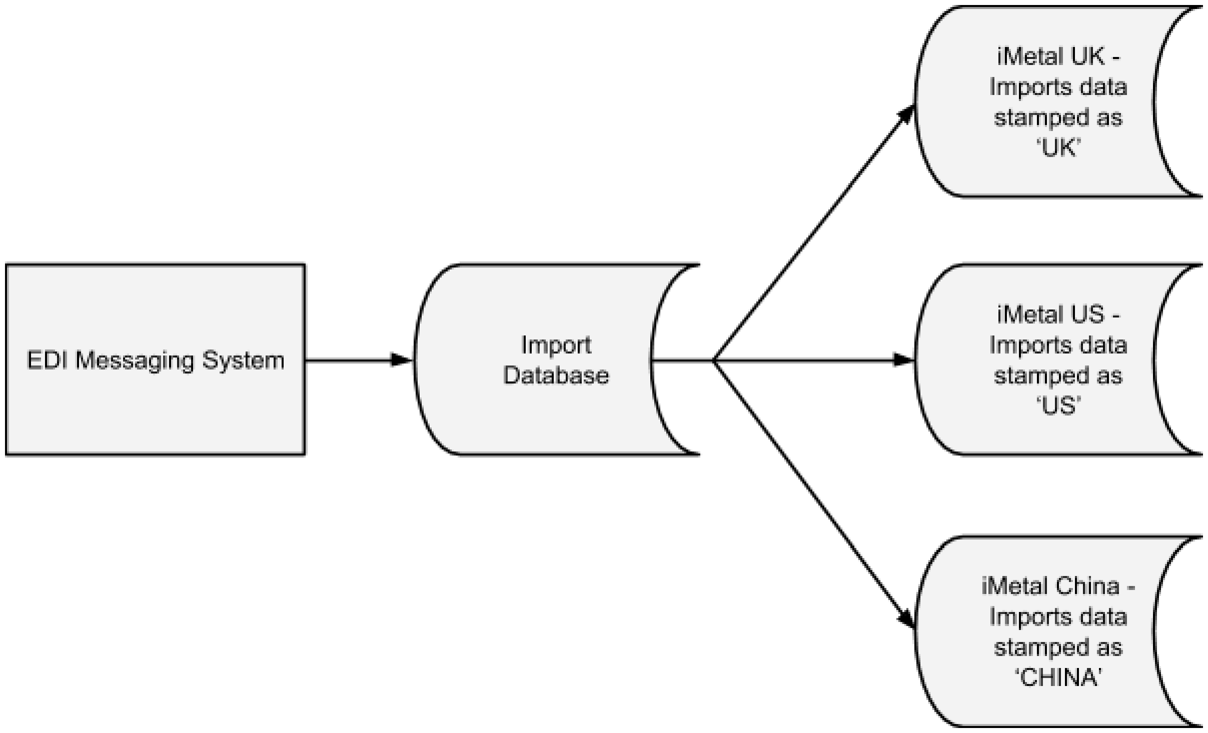
3. Import Database

Generally, all of the data tables required by a single iMetal installation are stored in a common database (e.g, Company A and Company B running on the same server would each have their own iMetal databases). However, the iMetal importing applications will be developed so that they can point to a common database in order to access the import data.

Whilst this won't be necessary in most situations, it does simplify the development of external systems that export data to multiple iMetal sites. It also provides an additional layer of security, in that external systems can be given direct write access to the import database alone, without also giving them access to the full iMetal database itself.

3.1 EDI Example

In the following example an EDI messaging system, which might receive data that is destined for three different iMetal systems within the same organisation, doesn't have to open three different database connections when passing it on to iMetal. It can simply write to the central Import database, specifying a 'destination' against each record. Each different iMetal installation will then be responsible for processing the import data


EDI Messaging System
Import Database
iMetal UK - Imports data stamped as 'UK'
iMetal US - Imports data stamped as 'US'
iMetal China - Imports data stamped as 'CHINA'

3.2 New Database Connection

The stock allocation import will use the same common database connection that has already been developed for the Sales Document Import application:

3.2.1 Additional Properties

The 'emetal.properties' file has been updated to include a new section as follows:

```
importdb.driver=org.postgresql.Driver
importdb.url=jdbc:postgresql://host:11061/imetal_imports
importdb.username=base_emetal
importdb.password=password
importdb.hibernatedialect=org.hibernate.dialect.PostgreSQLDialect
importdb.showsql=false
```

This is the standard configuration, which assumes that the database will use Postgres, but it could be configured to use other supported database systems too.

3.2.2 New Service

In order to communicate with this new database, a new service has been added to the iMetal server daemon:

```
Interface:
com.metalogicplc.emetal.client.services.ImportService

Implementation:
com.metalogicplc.emetal.client.services.impl.parallel.ImportServiceImpl
```

3.2.3 Spring/Hibernate Configuration

The Spring configuration files will have also been updated to include the necessary wiring for them.

The two configuration files involved are:

```
emetal.server\emetal.serverdaemon\src\main\resources\application.xml
emetal.server\emetal.httpservlet\src\main\resources\httpservlet-servlet.xml
```

4. Stock Allocation Table Structure

Within iMetal itself, stock allocations can be made to a number of different objects:

- Sales Items
- Process Requests
- Process Groups
- Standard Production Jobs
- Transport Plans
- Despatch Notes
- Invoices

In the initial release of the stock allocation import, only allocations against Sales Items will be supported (although allocations will automatically be created against process requests on sales items).

4.1 Import Header Fields

At the top of the main Stock Allocation import table is a header section containing a number of key fields:

Field Name	Type	V/M	Notes
import_company_reference	varchar(30)	M	e.g. UK, US, CHINA
import_batch_number	integer	M	Reference to import batch
import_status	character(1)	VM	(E)ntered, (F)ailed, (I)mported
import_notes	text		Leave blank
import_user_name	varchar(50)	MV	
import_source	varchar(50)		Source of import data
import_date	date		Leave blank
import_action	char	MV	(A)dd, (C)hange, (D)elete

4.1.1 Import Company Reference

This field is primarily used when multiple iMetal systems are configured to use the same import database. Each iMetal system will use the following server preference to determine its own company reference, which will then be used to find only the import data that is relevant to it:

```
Preference  : ImportCompanyReference
Default     : null

Description :
Determines the Company Reference for this iMetal system when looking for data in
the import database.
```

4.1.2 Import Batch Number

This numeric reference can be used to group multiple allocations together into a single common ‘import batch’. The reference will also be stamped onto any allocations created within iMetal and made available for searching and/or viewing in selected applications.

4.1.3 Import Status

This field determines the current status of each stock allocation being imported. It will start as Entered for each newly created stock allocation, but will then be set to either Failed or Imported during the update.

For some import failures, the problem may be resolved by either creating/amending reference data within iMetal or by altering the import records directly. The user may then alter the status back to Entered so that the allocation becomes eligible for importing again.

Note that if a record with a status of Imported is changed back to Entered, the system will attempt to import it again, but will fail during the validation process.

4.1.4 Import Notes

This field is available on each of the import tables and is used by the import validation routine whenever one or more fields fail their validations. It is an unlimited size text field, and will contain details in the following format:

```
branch_code: “COV” not found in branches table.
customer_code: “CLA002” not found in companies table.
exchange_rate_type_code: “6” must be one of “1”, “2”
```

The import program will automatically clear down the field at the start of each validation, in case it is being re-processed after data has been corrected.

4.1.5 Import User Name

This field will be validated against the ‘personnel’ table in iMetal and can be used as a selection on the import application.

It is important to note that this field is *not* used by the import program to read in group preferences. They are based on the user that is running the import itself.

4.1.6 Import Source

This optional field can be used to define the source of the import data (e.g. “iMetal Company A”, “Customer B EDI”, etc). It will be stamped onto the iMetal stock_allocations table.

Although this is a free-format field, any implementations with multiple import sources should carefully consider the format to be used so that reporting/querying of the data is straightforward.

4.1.7 Import Date

The import date field will be stamped onto the import header once the allocation has been successfully imported. The primary purpose behind this is to allow future archiving utilities to delete all imported allocations older than a certain date.

4.1.8 Import Action

This field will determine whether the import program treats the stock allocation data as a new record (Add), a change to an existing record (Change) or a deletion of an existing record (Delete).

5. Import Table Definition

This section describes the import table that needs to be populated with data before iMetal can import a stock allocation.

In the table, the ‘V/M’ column indicates whether the field is Validated (V) and/or Mandatory (M) or Defaultable (D). Where fields are mandatory in iMetal, but can be defaulted based on other data, these will be marked as ‘D’.

5.1 Stock Allocation Layout

The stock allocation table contains all of the information required in order to import a stock allocation into iMetal. No other tables are used.

Table Name: import_stock_allocations

Index 1 (Not Unique):
import_company_reference, import_batch_number, stock_branch_code, stock_item_number

Index 2 (Not Unique):
import_company_reference, import_batch_number, import_status

Index 3 (Not Unique):
import_company_reference, import_status

Field Name	Type	V/M	Notes
import_company_reference	varchar(30)	M	e.g. UK, US, CHINA
import_batch_number	integer	M	Reference to import batch
import_status	character(1)	VM	(E)ntered, (F)ailed, (I)mported
import_notes	text		Leave blank
import_user_name	varchar(50)	MV	
import_source	varchar(50)		Source of import data
import_date	date		Leave blank
import_action	character(1)	MV	(A)dd, (C)hange or (D)elete
automated_process_type	character(1)	V	(N)one, (R)apid Cutting, (P)icking
allocation_type	character(1)	MV	(S)ales Item, (D)espatch Item, (I)nvoice Item, (J)Standard Production Job, (P)rocess Group, (R)Process Request (1)
allocation_branch_code	varchar(6)	MV	Linked to ‘branches’ table (2/3)
allocation_import_number	integer		Used to find original import document if external system doesn’t know the iMetal number (e.g. if sales order was imported).(3)
allocation_header_number	integer	V	(2)
allocation_item_number	integer	V	(2/3)

stock_branch_code	varchar(6)	MV	Linked to 'branches' table
stock_item_number	varchar(16)	MV	
allocated_pieces	integer	MV	
allocated_quantity	numeric(12,3)	MV	
allocated_weight	numeric(10,3)	MV	
comments	text		
firm	boolean	D	Defaults to false
packing_weight	numeric(10,3)		For future use
invoice_weight	numeric(10,3)		For future use
composite_key	varchar(100)	M	Unique string value comprised of the import_company_reference + import_batch_number + allocation_branch_code + allocation_import_number + allocation_item_number e.g. US-24-HOU-4833-1

Notes:

(1) The Allocation Type field will be used to determine what kind of allocation is required. Only Sales Item allocation are supported in this release.

(2) These three fields (allocation_branch, allocation_header_number and allocation_item_number) will be used along with the Allocation Type to find the record that the stock item needs to be allocated to. For example, if the Allocation Type is 'S', the three allocation fields will represent the Sales Order Branch, Sales Order Number and Sales Item Number.

(3) These three fields (allocation_branch, allocation_import_number and allocation_item_number) will be used along with the Allocation Type to find the record that the stock item needs to be allocated to. For example, if the Allocation Type is 'S', the three allocation fields will represent the Sales Order Branch, Import Number and Sales Item Number. The import program will then try to find the iMetal sales item record with this combination of values. These will be used for companies that import both the sales orders and stock allocations, where they won't know which sales order number has been assigned by iMetal.

5.1.1 Automatically Populated Internal Stock Allocation Fields

The following internal fields will be populated automatically by the import program. These are fields in the iMetal stock_allocations table and not the import table itself:

Field Name	Type	V/M	Notes
product_id	integer		From stock item's product id
warehouse_id	integer		From stock item's warehouse id
dim1 to dim5	numeric(9,4)		From stock item
reserve_only	boolean		false
expiry_date	date		null
parent_id	integer		null
item_type	char(1)		Based on allocation_type (1)
sales_item_id	integer		Link to sales item (2)
despatch_item_id	integer		Link to despatch note item (3)
job_consumption_id	integer		Link to standard production job (4)
inbound_allocation_id	integer		null
vehicle_stop_detail_id	integer		null
product_level_allocation_id	integer		(5)
printed_date	date		null
inbound_allocated_pieces	integer		null
inbound_allocated_quantity	numeric(10,3)		null
inbound_allocated_weight	numeric(12,3)		null

Notes:

(1) The allocation_type from the import will be mapped to the item_type field as follows (note that Sales Item types will be supported initially):

- (S)ales Item = (**P**)roduct level allocation (see below)
- (D)espatch Item = (**D**)espatch item
- (I)nvoice Item = (**S**)ales item
- (J)Standard Production Job = (**M**)
- (P)rocess Group = (**P**)roduct level allocation (see below)
- (R)Process Request = (**P**)roduct level allocation (see below)

(2) The id of the invoice item that the allocation is being made against. Not supported initially.

(3) The id of the despatch item that the allocation is being made against. Not supported initially.

(4) The id of the standard production job that the allocation is being made against. Not supported initially.

(5) The product level allocation id that the allocation has been created against. See Product Level Allocations section below.

5.1.2 Finding the Allocation Record

Each stock allocation that is imported will need to be allocated against something (e.g. sales item, production job, invoice, etc) so the import program will need to use the allocation_* fields to find the relevant record.

For Sales Item type allocations, the logic will be as follows:

1. If the allocation_branch_code, allocation_import_number and allocation_item_number fields have been provided, the program will attempt to find a Sales Order record in the sales_headers table using the branch id and the import number (see appendix C for details of the new import_number field on the sales headers table). If no order is found, or if the order status is not Entered (the internal status part), then the import will fail with a validation error. If the header is found, then the program will attempt to read in the appropriate item using the allocation_item_number and the sales header id. Again, if the item is not found or its status is not Entered, then the import will fail with a validation error.
2. If the allocation_branch_code, allocation_header_number and allocation_item_number fields have been provided, the program will attempt to find a Sales Order record in the sales_headers table using the branch id and the header number. If no order is found, or if the order status is not Entered (the internal status part), then the import will fail with a validation error. If the header is found, then the program will attempt to read in the appropriate item using the allocation_item_number and the sales header id. Again, if the item is not found or its status is not Entered, then the import will fail with a validation error.

Note: When checking the status of a sales item, it is the ‘internal status’ that must be Entered, not the status itself.

5.1.3 Product Level Allocations

If the Item Type on the import record is either Sales Item, Process Group or Process Request, the stock_allocations record itself will point to a Product Level Allocation, rather than directly to the sales item, process group or process request.

For sales item type allocations, the logic described above will be used to find the sales item that the allocation is for. This item will then have its own product level allocation, against which the stock allocation can be created.

However, rather than simply using the product level allocation from the item, the code will first look to see if there are any Process Requests against it. If there are, then the program should loop around them, looking for one with an Allocation Balance Quantity greater than zero. If one is found, then that product level allocation should be used. If all of the process requests are fully allocated, then the original product level allocation from the sales item will be used instead.

The code will *not* attempt to split the stock allocation across multiple process requests.

6. Import Validation

There are two key goals to the validation step:

1. To avoid importing incorrect data into iMetal.
2. To provide information on all of the import failures back to the exporting system.

6.1 Field Validation

Each of the import tables in this document highlights which fields need to be validated and under what circumstances. In all cases, a validation failure will result in the import of the entire allocation being abandoned, but not before all of the fields on all of the import tables have been validated. For example, if the validation reads in the allocation and finds that the Branch Code doesn't exist, the import program will still continue to validate all of the other fields on the import record. In that way, a user examining the results of the validation will be able to resolve all of the problems at once, without having to keep trying the import after each correction.

When a field validation fails, the import program will set the 'import_status' field on the allocation import record to Failed and will add a description of the failure to the 'import_notes' field of the table being validated. This is an unlimited size text field, so the result will be something like the following:

```
branch_code: "COV" not found in branches table.
customer_code: "CLA002" not found in companies table.
allocation_type: "Z" must be one of "S", "D", "I", "J", "P" or "R".
```

The import program will always use the same format for the message:

```
Table Validations:
field_name: "import value" not found in reference_file table.

Enumeration Validations:
field_name: "import value" must be one of "options list".
```

6.1.1 Quantity Validations

The import record contains three quantity fields; Pieces, Quantity and Weight, which the import validation program will pass through the standard quantity calculation routine using the product, dimensions and dynamic density of the stock item. This will populate any missing values, allowing the import program to just specify Pieces and have the Quantity and Weight calculated automatically.

After performing this calculation, the validation routine will then check to make sure that none of the three values exceed the Free quantity on the stock item. If they do, then the validation will fail and the import will be aborted (after validating any remaining fields on the item).

6.2 Batch Validation

The validation will take place against each stock allocation *within* a batch, which means that if ten allocations are received in the batch and one of them fails the validation, the remaining nine allocations will still be imported.

6.3 Successful Import

If an allocation is successfully imported into iMetal, the ‘import_status’ field will on the header will be set to Imported. It will also update the ‘import_notes’ field on the header with a reference to the allocation that has been created in iMetal:

```
Examples:
Stock Allocation Created Against Sales Item: COV-123441-3
Stock Allocation Created Against Despatch Item: LON-48343-3
Stock Allocation Created Against Process Request: COV-39245
```

Note that only sales item types will be implemented initially.

7. Triggering Imports

Two different interfaces will be provided for triggering the import of allocations; a Selection Panel and a Command Line Interface.

7.1 Stock Allocation Import Panel

The Stock Allocation Import panel provides users with the ability to manually trigger the validation and/or importing of allocations and will be available as a normal application on the iMetalImports sub menu.

7.1.1 Import Selection Panel

The import panel will provide the following selections:

- Import Company Reference (defaults from ImportCompanyReference preference and is not editable by the user)
- Import Batch Number Range
- Import Status (dropdown with Choose..., Entered and Failed as options). The import will never select documents with a status of Imported.
- Import User (dropdown selection using Personnel table)

7.1.2 Buttons

The following buttons will be available:

- Validate Only - perform a validation pass on import data matching the selections
- Import - validate and import data matching the selections
- Clear - clear the selections
- Help - link to the help wiki page

Validate
The validate button will loop around all of the import stock allocations matching the selection criteria and perform an import validation on each one. If any of the validations fail, the program will flag the record as Failed and write any failure details to the import record(s) involved.

Once the validation is complete, a dialogue will be displayed showing the results:

```
                    Stock Allocation Validation Results

Documents Failing Validation:   99999
Documents Passing Validation:   99999

Total Documents:                99999
```

Import

The import button will loop around all of the import stock allocations matching the selection criteria and perform an import validation on each one. If any of the validations fail, the program will flag the record as Failed and write any failure details to the import record(s) involved. If all of the validations on an allocation pass, the program will then import it into iMetal.

Once the import routine is complete, a dialogue will be displayed showing the results:

```
                    Stock Allocation Import Results

Documents Failing Validation:   99999
Documents Imported:             99999

Total Documents:                99999
```

7.2 Command Line Import

A command line connection to the import routine will be developed, which will allow an import run to be performed either manually on the command line or via automated scripts or 'crontab' entries.

The command line interface will call the same routine as the manual screen defined above, but with just three optional parameters:

```
BatchNumberFrom=999999
BatchNumberTo=999999
ImportUser=Steve_Baker
RunType=VALIDATE or IMPORT

NOTES:
If neither BatchNumber parameter is present, the import will find all imports
regardless of batch number.

If only BatchNumberFrom is present, it will find all imports with that batch
number or higher.

If only BatchNumberTo is present, it will find all imports with that batch
number or lower.

The ImportUser option must have any spaces replaced by underscores (e.g. "Steve
Baker" appears as "Steve_Baker") in order to successfully pass the parameter
into iMetal.  iMetal will then turn the underscores back into spaces.

The RunType parameter will determine whether the program simply validates the
data or validates and imports it.  If the parameter is not present, it will
default to IMPORT.
```

Internally, the import program will only select imports with a status of Entered. It will also only select imports where the import_company_reference field matches the value of the "ImportCompanyReference" server preference.

After completion, the import routine will display the results as a simple text output:

VALIDATE:

```
VALIDATION
FOUND   : 99999
FAILED  : 99999
PASSED  : 99999
TOTAL   : 99999
```

IMPORT:

```
IMPORT
FOUND   : 99999
FAILED  : 99999
IMPORTED: 99999
TOTAL   : 99999
```

Appendix A - Technical Details

A.1 - Allocation Creation Options

There are two possible approaches to developing the import routine, each of which has its own pro's and con's:

1. Use the existing Service methods that are called by the stock allocation panel.
2. Develop a brand new application to create the stock allocation.

A.1.1 Existing Methods

The main method for creating stock allocations when called from the client is SaveStockAllocationControl.saveStockAllocationsDTO, which will take a DTO and create the necessary stock allocation from it.

An import application could be written which behaves like an automated StandaloneStockAllocationEntryPanel application; populating the relevant DTO using the data from the import table, then calling the above method to persist the allocation. It would first validate all of the data, then turn it into a DTO before calling the method.

The key benefits of this approach are:

- Less development required
- Future changes to the normal save process will also be incorporated in the import

Downsides are:

- May not run as fast as a dedicated import routine
- The process will not be quite as simple as described above. The code will need to perform various operations on each imported object in order to fully populate the DTO objects (e.g. performing quantity calculations). However, this is also true of a dedicated import routine

The rest of this document assumes that the 'Existing Methods' technique will be used to develop the import.

A.1.2 New Application

Rather than making multiple calls to existing services using DTO's, a brand new application could be developed, which converts the imported data directly into Stock Allocation domain objects and then persists them on the database.

The key benefits of this approach are:

- May provide a more tailored application, which runs more efficiently and quickly

Downsides are:

- Development and testing may require more time than reusing existing code
- The import routine and standard stock allocation creation routine are at risk of divergence, as new features may need to be added to both

Appendix B - Add, Change, Delete

This section describes the behaviour of the import routine as it processes documents for each *Action* type; Add, Change or Delete.

B.1 Add Document

The descriptions in the rest of this document and particularly in Appendix A, are centered around the addition of brand stock allocations to iMetal via the import program, so no further details are required here.

B.2 Change Document

When an import stock allocation is found with an import_action of ‘C’ for Change, the import routine will still validate the data in exactly the same way as is does when adding new documents. Instead of creating a new stock allocation in iMetal, however, the program will attempt to find the allocation owner and the allocation itself and then make (limited) changes to it based upon the data in the import tables.

If the allocation is found and *it has no child allocations*, then the program will attempt to alter the Allocated Quantities to the new values specified in the import record. Existing methods for amending allocation quantities can be re-used for this.

If the allocation does have child allocations (e.g. it was originally allocated to a sales item, but is now on a transport plan) then the import will be aborted and an appropriate validation message added.

B.3 Delete Document

When an import stock allocation is found with an import_action of ‘D’ for Delete, the import routine won’t validate the data in the same way as is does when adding new documents. Instead, it will only validate the following fields on the ‘import_stock_allocations’ table:

Field Name	Type	V/M	Notes
allocation_type	character(1)	MV	(S)ales Item, (D)espatch Item, (I)nvoice Item, (J)Standard Production Job, (P)rocess Group, (R)Process Request (1)
allocation_branch_code	varchar(6)	MV	Linked to ‘branches’ table (2)
allocation_import_number	integer		Used to find original import document if external system doesn’t know the iMetal number (e.g. if sales order was imported).
allocation_header_number	integer	V	(2)
allocation_item_number	integer	V	(2)
stock_branch_code	varchar(6)	MV	Linked to ‘branches’ table
stock_item_number	varchar(16)	MV	

If these validations pass, the program will then attempt to find an existing allocation by first finding the owner of the allocation, then finding the stock allocation itself using the stock branch and stock item number.

If an allocation is found, the program will then validate to decide whether it can be deleted by checking to see if it has any Child Allocations (e.g. it was originally allocated to a sales item, but is now on a transport plan).

If the allocation is found and has no child allocations, then the import program will delete the stock allocation. Existing methods for deleting stock allocations can be re-used for this.

Appendix C - Existing Database Changes

The following changes will be required to existing iMetal tables.

C.1 Sales Headers

The following new fields will be added to the sales_headers table:

```
New Field:
import_number integer;

New Index:
NOT UNIQUE: branch_id, type_id, import_number
```

This change is required in order to allow the stock allocation import routine to find the sales order involved when the external system only knows the *old* order number and not the new one. For example, if the sales order import routine was used to import orders from an old external system before go live and then the stock allocation import routine was used to import the allocations for those orders.

In order to populate the field, a change will need to be made to the Sales Order Import routine so that it sets the value from the import_number field on the import_sales_headers table.

The field will be added as an optional column to the Sales Header Enquiry application.

It will also be added as view-only field to the References tab of the Sales Document Entry application, to the right of the existing Contract and Release Number fields.

Appendix D - Data Import Options

There are two options for populating the import tables with data.

D.1 Direct Database Connection

Using either a direct database or ODBC connection, an application could write directly to the import tables using standard SQL commands.

This is a relatively straightforward process, especially since there is only a single import record for each stock allocation, which means that there is no chance of the import routine picking up an import record where the child records have not yet been created (unlike with Sales and Purchase imports).

D.2 CSV Import

This method can be used for exporting systems that have no ability to write to SQL databases. The import table layouts described above are still used, except that they are flattened into comma-separated format. All of the notes above are still relevant, therefore.

D.2.1 Basic Rules

The basic rules of the CSV import file are as follows:

- One CSV file is required for each import table (there is only one in the case of Stock Allocations)
- Each field is separated by a comma.
- Where text fields contain single or double quote characters, they must be preceded with a backslash. For example; \” or \’
- Where a comma is present in the text, a backslash escape character must precede it: \,
- Empty fields can be left completely empty, so "1,,3" would import values of '1', null and '3'.
- Each record is separated by a single line feed character (ASCII 10, hex 0a).
- Linefeed characters within text are represented by the \n escape sequence.
- Each field in the import file must be presented in the exact order defined by the layout above.
- Every field in the above layout must be present in each line of the import file.
- Date fields must be in the format CCYY-MM-DD (e.g. 2014-08-28).

D.2.2 Import Process

In order to import CSV data into the iMetal import tables, the following steps are required *for each import table*. The import_stock_allocations table is used in this example:

```
Copy the import file to /usr2/base_emetal on the destination server

Log on to to the server as base_emetal

psql import_imetal    (name of import database)

Type:
COPY import_stock_alloactions FROM '/usr2/base_emetal/import_file_name DELIMITER
',' null AS ;

If the command is successful, then a response should be given saying "COPY n"
where n is the number of records imported.  Note that the only validations
performed at this stage are to make sure that all the fields are present and
that each field is of the correct type.  The import will stop at the first
failure.

To list the imported records:
select * from import_stock_allocations;
```

Appendix E - Test Plan

In order to comprehensively test the import functionality, the following checks will need to be performed:

- Create a sales order and sales items using the standard order entry application:
  - Item 1 should have no allocations
  - Item 2 should have a single process request
  - Item 3 should have two process requests
- Create a sales order and single sales item using the Sales Import application
- Create import stock allocations for each of the above items and make sure that they are allocated to the correct place each time (either directly to the item, or to the process request).
- Check the product balances before and after the allocations to make sure that they are correct
- Check the stock allocation quantities to make sure they are correct
- Check the Stock Allocation Enquiry to ensure that they are listed correctly
- Make sure that each stock allocation can be amended and deleted using the normal iMetal Stock Allocations panel
- Create *Change* type import stock allocation records that will modify the allocated quantities
  - Use both the allocation_import_number and allocation_header_number methods to find the existing allocation
- Create *Deletion* type import stock allocations that will delete stock allocations. Try this with stock allocations created using the import and also allocations created manually
- Every validated field must be tested using data that should fail to ensure that the import is blocked and that the reason is reported correctly

Appendix F - Functionality Not Covered

The following functionality is *not* covered in this development:

- The following allocation types will not be developed initially:
    - Despatch Items
    - Invoice Items
    - Standard Production Jobs
    - Process Requests (directly)
    - Process Groups
- Enquiries or reports listing import table data
- Maintenance programs for each import table
- Deletion of successfully imported data (1)

(1) The indexes on each import table have been designed to allow import data to be stored indefinitely without impacting on the performance of the import search routines.