HP
EDI850 User Defined Information usage for Configuration Services

v1.1

February 28, 2014

Note

This document was allocated the following unique HP part-number: **468514-104**

It is therefore always possible to access the latest version of it from the HP Supplier Portal.

For more convenience, changes versus the previous document revision are highlighted.
You can access a version with changes NOT highlighted using the HP part-number **468514-103**.

Table of Contents

1. General introduction 5
1.1. Preliminary remarks 5
1.2. Prerequisites 5
2. Introducing the User Defined Information (UDI) 6
2.1. Important warning related to the EDI850 “Element delimiter” 6
2.2. Rules behind UDI’s in EDI850 6
2.2.1. Preliminary remarks 6
2.2.2. Terminology 7
2.2.3. Location of UDI in the EDI850 file 7
2.2.4. The “N9” Segment combined with the “ZZ” Qualifier 8
2.2.5. The “MSG” Segment(s) following the “N9” Segment with “ZZ” Qualifier 8
3. UDI usage by Configuration Services 10
3.1. Configuration Services in scope 10
3.2. Detecting orders with UDI-dependent services 10
3.2.1. Two possible implementations 10
3.2.2. Specificities for Non-UPS-enabled factories 10
3.2.3. Specificities for UPS-enabled factories 11
3.3. An example of Configuration Service deliverable 11
4. How to parse the EDI850 UDI lines 12
4.1. The EDI850 UDI data structure 12
4.2. Configuration Services to only rely on Structured UDI lines 13
4.3. Recommended way to parse the UDI section provided via EDI850 14
4.4. Checking the validity of the Invoice UDF data values 14
4.4.1. Non-UPS-enabled factories 15
4.4.1. UPS-enabled factories 15
5. Pilot Plan for the associated capability 17
5.1. Pilot Contents 17
5.2. Pilot Timeline 17

6. Examples 18
6.1. Short structured UDI lines (Invoice UDF's) – Valide Case 18
6.2. Several unstructured UDI lines – Valid case 19
6.3. 1 unstructured and 1 structured line – Valid case 19
6.4. 1 unstructured and 1 structured line – Invalid case 20
6.5. 1 unstructured line being interpreted as being structured 20
7. Document maintenance / Feedbacks 22

Glossary

Configuration Service	One service from the "CS" portfolio of services
Configuration Services	The HP Portfolio of Services executed at factory level Used to be called "CIS"; sometimes shortened to "CS"
Factories	Generic term for HP Manufacturing Partners
CS	Short name for "Configuration Services"
Invoice UDF	Invoice User Defined Field A structured type of UDI, located at Order Header level
UDI	User Defined Information associated to an order
UUT	Unit Under Test A unit being produced on the factory's production floor
UPS	Unit Personalization System An HP-owned tool in charge of Unit Personalization tasks

1. General introduction

1.1. Preliminary remarks

This specification primarily targets HP Manufacturing Partners (or 'factories') which EDI850 data flow was enhanced *upfront* with the "User Defined Information" (UDI).

Other factories could of course still be interested by the reading of this specification, especially if there is a plan in place to soon enhance their EDI850 data flow with UDI.

The main purpose of this document is actually:

1. To introduce factories to the UDI fields in EDI850 data flow
2. To explain *in which Configuration Services cases* the factories may have to retrieve one or more UDI from the EDI850 data flow
3. To explain, in the aforementioned cases, *how* factories are expected to make use of these UDI, in order to end up providing the right piece of information to the associated services

Note that this specification results from a *Regional EMEA initiative*, and that it is not confirmed yet (at the time of writing) that the same principles and technologies would be used at WW level, in case a similar effort was to be initiated by WW teams in the future.

In order to illustrate the requirements detailed in the next pages with practical examples, we will use –whenever it makes sense– the MATINF03 (also abusively called "CNRS") deal case.

1.2. Prerequisites

Before a Factory can actively start working onto the local implementation of the present specifications, a shared HP/Factory IT-side initiative must have been completed upfront, in order to ensure that the UDI is propagated to the Factory via EDI850.

It is also required to ensure, as another prerequisite, that the UDI –beyond being delivered to the factory through EDI850– is also captured by the Factory and recorded into the local SAP database.

Factories can decide though between recording in the local database the whole UDI section in EDI850, and *delay* the actual parsing to a later point of time – or to perform the parsing *right away*, in order to avoid storing UDI's which will prove to be useless for the local fulfillment of the order.

2. Introducing the User Defined Information (UDI)

2.1. Important warning related to the EDI850 “Element delimiter”

The EDI850 specification defines several “delimiters”, in order to give a structure to the otherwise flat text file.

One of these delimiters is called the Element delimiter: this is the delimiter that separates the various elements of a given segment (as well as the segment).

For instance, here is a typical “Transaction Set” segment at EDI850 File Header level, with its 2 elements (for the purpose of the explanation, the element delimiter is represented as <Element delimiter>):

```
ST<Element delimiter>850<Element delimiter>0001
```

Depending on the targeted factory, the actual character used as Element delimiter can vary.

The most typical common character for Element delimiter is the *tilde* character (“~”).

However, for some factories, the *asterisk* character (“*”) is known to be used as Element delimiter instead of the *tilde* character. And in some other cases, *yet another* character could be used…

For the sake of simplicity, we will do -over the remaining pages of this document- *as if* the *tilde* character was the shared, single Element delimiter used across all factories.

The above example will therefore be provided as:

```
ST~850~0001
```

In case, for instance, the *asterisk* character is used for a particular factory, the related readers will have to properly translate it into:

```
ST*850*0001
```

2.2. Rules behind UDI’s in EDI850

2.2.1. Preliminary remarks

Only a small portion (a few percent maximum) of all orders do contain UDI’s.

The orders *without* UDI will show up in EDI850 files *exactly the way* they used to show up *before* the factory’s EDI850 data flow was enhanced to include UDI.

2.2.2. Terminology

In order to avoid misunderstandings over the following pages of this document, it is important to carefully define a *terminology* for the various consecutive steps along the UDI "lifecycle".

First of all, the User is allowed, at Order Entry level, to associate *several* independent UDI's to his order.

In the simple example that follows, the User entered for instance 3 independent UDI's:

```
This is my first independent piece of information.
This is my second independent piece of information.
This is my third independent piece of information.
```

Whenever referring to the overall User Defined Information entered *at Order Entry level*, we will use the term **Native UDI**.

An *independent* piece of information defined by the User at Order Entry level will be called a **Native UDI line**.

Whenever referring to the overall User Defined Information provided to the factories *through EDI850 files*, we will use the term **EDI850 UDI**.

To keep consistency with the Native UDI terms defined above, an *independent* piece of information defined by the User at Order Entry level and *translated into the EDI850 format* will be called an **EDI850 UDI line**.

Since, as we will see, a long EDI850 UDI line can spread over several consecutive EDI "MSG" segments, we need to carefully name the information found into *each of these segments* – which otherwise could be considered as an "EDI850 UDI line".

Even though the "MSG segments" are used for other reasons than just UDI in EDI850, we will call **MSG Segment data** (no need to explicitly refer to EDI850) any UDI piece of information stored in *one* MSG segment.

Lastly, we need to have a term to qualify the UDI's *after they have been parsed by the factory* (from the EDI850 files). These UDI's will be called **Parsed UDI**.

And here again, we will use the term **Parsed UDI line** to name an independent piece of information entered by the User, as "rebuilt" by the factory after the parsing phase occurred.

Whenever though the context defines in a non-ambiguous way *which* UDI we are talking about (Native vs EDI850 vs Parsed), we will just make mention of the generic "UDI" term.

2.2.3. Location of UDI in the EDI850 file

First of all, the UDI in our scope applies to *a whole order* – not *order items* individually.

Factories must therefore expect the UDI of a given order to be located at the Order Header level. Typically, the UDI could be located e.g. between the order number related details ("REF" segments) and the customer name related details ("N1" segments).

HP recommends though to NOT hardcode the location of UDI into the EDI850 Order Headers, and to rather implement it in a dynamic way (based upon the Segment keyword).

2.2.4. The "N9" Segment combined with the "ZZ" Qualifier

The actual UDI in an order get introduced by a single "N9" Segment, exposing the dedicated "ZZ" Qualifier, followed itself by the hard-coded "User Defined Information" text – as follows:

```
N9~ZZ~User Defined Information
```

Note that an "N9" Segment must normally be followed by one or more "MSG" Segments. However, in the event that it is NOT followed by any "MSG" Segment, factories are invited to conclude that the current order actually does NOT have any UDI.

2.2.5. The "MSG" Segment(s) following the "N9" Segment with "ZZ" Qualifier

"MSG" Segments are used already today in some factories, to provide (Shipping, Picking, Delivery, Export, etc.) instructions to the factory.

"MSG" Segments are *also* used to host the UDI – as follows:

```
MSG~<MSG Segment data>
```

There can be *one or more* consecutive "MSG" Segments after an "N9" Segment with "ZZ" Qualifier.

Here is an example with 5 "MSG" Segments:

```
MSG~<MSG Segment data #1>
MSG~<MSG Segment data #2>
MSG~<MSG Segment data #3>
MSG~<MSG Segment data #4>
MSG~<MSG Segment data #5>
```

Factories must NOT assume any maximum amount of "MSG" Segments for the UDI of a given order. There could theoretically be *hundreds* of such segments per single order. In practice though, there would be only *a few* segments (e.g. 12 segments being already a lot).

It is HP's responsibility to ensure that the MSG Segment data values will never contain any of the following forbidden characters:

- "|" (*pipe* sign, ASCII code 124)
- "~" (*tilde* sign, ASCII code 126)
- "*" (*asterisk* sign, ASCII code 42)
- "<" (*inferior* sign, ASCII code 60)
- ">" (*superior* sign, ASCII code 62)

Factories are ONLY expected to parse the EDI850 UDI lines when a Configuration Service AV (or SKU) in the order depends on one or more UDI's. More details related to how factories shall detect the aforementioned condition can be found in Chapter 0 – while the Chapter 4 will focus on how exactly to parse the UDI lines.

In all other cases, factories can just skip the EDI850 UDI lines.

3. UDI usage by Configuration Services

3.1. Configuration Services in scope

The best Configuration Service "candidate" to UDI is the **HP Standard Asset Tagging Service** (AY111AV). However, we can consider that *any* Configuration Service could be defined as having a dependency onto one or more UDI values.

3.2. Detecting orders with UDI-dependent services

3.2.1. Two possible implementations

Factories can decide to either:

- *Systematically copy all EDI850 UDI lines* into their local Order Management database, and parse them from there when required by the local Configuration Services process – *or…*
- Check *during the EDI850 parsing phase*, for each order received, if a Configuration Service in the order would require a UDI, and only store in the local Order Management database the UDI's that will be used by Configuration Services

3.2.2. Specificities for Non-UPS-enabled factories

Factories having no UPS capability in production will need to manage their own list of Order Items having a dependency onto UDI's.

Such Order Items can be of 2 different kinds:

1. **Customer-Specific AV's** (e.g. YH318AV)
   - *This case applies to orders in CTO mode*
   - In most cases, *only 1 AV* in the order will have got a dependency onto one or more UDI's. However, factories are invited to push further the CTO approach, and be ready to support more extreme cases where *2 or more AV's* in the order have got a dependency onto UDI's.
2. **Complex SKU's** (e.g. G4Z69EC)
   - *This case applies to orders in SKU mode*
   - Like in CTO mode, it is important to understand that there could be 2 or more services (Customer-Specific AV's) in the SKU with a dependency onto UDI's

In both of the above cases, the factory is now in a situation where it knows which (if any) Customer-Specific AV or AV's in the order have got a dependency onto UDI's.

Let's take the example of the MATINFO3 (a.k.a. "CNRS") deal, and let's name 'YT162AV' the *dynamic/multi-entity* Asset Tagging Service AV and 'F7D13AV' the *dynamic/multi-entity* Logo In Firmware Service AV.

Since the MATINFO3 deal is a pure CTO deal, the "list of Order Items having a dependency onto UDI's" would be made of 2 elements (assuming there is no other deal of this kind in production): YT162AV and F7D13AV.

Following the instructions provided in Chapter 4, factories would therefore have to detect all orders containing either -or both- of the aforementioned AV's, and retrieve from the related UDI the 2-letter "Service" information (as specified in the ATAN or CSAN deliverables).

3.2.3. Specificities for UPS-enabled factories

Factories running UPS will need to provide to the UPS process the expected UDI values (considered, in UPS terminology, as "OSI", i.e. "Order Specific Information" values), based upon the specific requirements of the service(s) in the order.

The UPS tool itself will only implements *generic* data names corresponding to "Invoice UDF" OSI's:

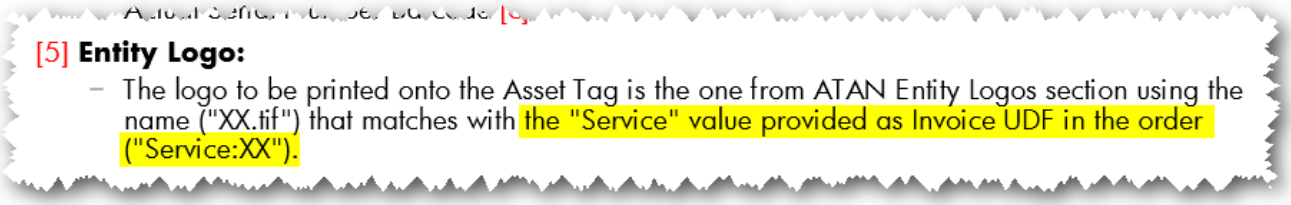
- **InvoiceUDF1** for the *first* Invoice UDF data the UPS service depends on
- **InvoiceUDF2** for the *second* Invoice UDF data the UPS service depends on (if any)
- **InvoiceUDF3** for the *third* Invoice UDF data the UPS service depends on (if any)
- etc… (as needed)

The *mapping* between these generic data names and the *actual Invoice UDF Keyword* which factories must use to retrieve the information from the EDI850 data flow isn't part of the UPS tool (since UPS is designed to manage all potential OSI's the exact same way).

It will be documented into the CS WW Engineering service deliverable. For instance, for Asset Tagging services, the mapping will be documented into the ATAN – in this case, in the PPT attachment, like the Chapter 3.3 illustrates it.

3.3. An example of Configuration Service deliverable

Using the MATINFO3 (or "CNRS") deal as an example, here is how a dependency onto a UDI would be expressed in a Configuration Service deliverable (here, an ATAN):


[5] **Entity Logo:**

- The logo to be printed onto the Asset Tag is the one from ATAN Entity Logos section using the name ("XX.tif") that matches with the "Service" value provided as Invoice UDF in the order ("Service:XX").

Note that, as we will see in Chapter 4.1, an Invoice UDF is a *structured* type of UDI: typically the type of UDI which the Configuration Services team would use for UDI-dependent services.

4. How to parse the EDI850 UDI lines

4.1. The EDI850 UDI data structure

Each MSG Segment data can be made of up to **70 characters**, not more.

However, a Native UDI line can be made of *more than 70 characters*. In this case, the overall string would be spread over 2 or more consecutive MSG Segments in the EDI850 file – as illustrated with this example:

Native UDI (wrapped for legibility purpose)

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.
```

Resulting data in EDI850 file

```
N9~ZZ~User Defined Information
MSG~Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusm
MSG~od tempor incididunt ut labore et dolore magna aliqua.
```

Here as well, there is no official maximum amount of characters for a single Native UDI line.

Indicatively though, a Native UDI line would in practice not spread over more than 10 consecutive "MSG" Segments in the EDI850 file (4 Segments being already a pretty rare case).

Of course, as said already, there could be *several* Native UDI lines – some potentially shorter than (or equal to) 70 characters, some potentially longer than 70 characters.

Here is a new example to illustrate this:

Native UDI (wrapped for legibility purpose, with end of lines represented as ■)

```
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua.■
Ut enim ad minim veniam■
quis nostrud exercitation ullamco■
```

Resulting data in EDI850 file

```
N9~ZZ~User Defined Information
MSG~Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusm
MSG~od tempor incididunt ut labore et dolore magna aliqua.
MSG~Ut enim ad minim veniam
MSG~quis nostrud exercitation ullamco
```

As an indication, here is how the Native UDI lines can be divided in terms of length:

- **Lines shorter than or equal to 70 characters** roughly represent 85% of all lines
- **Lines strictly longer than 70 characters** roughly represent 15% of all lines

4.2. Configuration Services to only rely on Structured UDI lines

Users (Customers) are left with the choice to give, or not, a *structure* to each of their Native UDI lines.

In case the User wants to assign a special role to a particular Native UDF line, he will want to give it a *structure*.

A *structured* Native UDI line is sometimes called an **Invoice UDF** (for Invoice User Defined Field).

An Invoice UDF is made of 3 distinct parts: a **UDF keyword**, a **UDF separator** and a **UDF value**.

It is possible to detect whether or not a given EDI850 UDI line corresponds to an Invoice UDF, by checking if it matches the following regular expression:

Regular expression to check if an EDI850 UDI line is an Invoice UDF

```
^\w+[^:]* *:{1,2} *.*$
```

This regular expression basically means that e.g.:

- The separator could be made of 1 or 2 consecutive colon (":") signs
- There could be spaces immediately before and/or immediately after the separator

In order to simplify the parsing of the information, and avoid error cases at factory level, the following rule has been defined:

Whenever used as part of a Configuration Service, an Invoice UDF will not exceed 70 characters.

Here are the regular expressions to be used to extract each part of an Invoice UDF:

Regular expression to extract the <u>UDF keyword</u> from an Invoice UDF (check the black parentheses)

```
~~^(\w+[^:])* *:{1,2} *.*$~~
^(\w+[^:]*) *:{1,2} *.*$
```

Regular expression to extract the <u>UDF separator</u> from an Invoice UDF (check the black parentheses)

```
~~^\w+[^:](* *:{1,2} *).*$~~
^\w+[^:]*( *:{1,2} *).*$
```

Regular expression to extract the <u>UDF value</u> from an Invoice UDF (check the black parentheses)

```
^\w+[^:]* *:{1,2} *(.*)$
```

Note that the UDF value, unlike the UDF keyword or UDF separator, may be empty.

4.3. Recommended way to parse the UDI section provided via EDI850

In summary, it is possible to apply the following basic algorithm in order to retrieve the right UDI piece(s) of information:

```
For each Order covered by the most recent EDI850 file
|   If the Order contains one or more services with a dependency on a UDI
|   |   For each MSG Segment associated to this Order
|   |   |   Read the next available MSG Segment data
|   |   |   If the data is an Invoice UDF
|   |   |   |   If the Keyword matches a Keyword used by a Service
|   |   |   |   |   Capture the associated data and get prepared to use
|   |   |   |   |       it for the execution of the related Service(s)
|   |   |   |   End If
|   |   |   End If
|   |   Next MSG Segment
|   End If
Next Order
```

Note that:

1. The above algorithm supports the extreme and unexpected case where 2 or more Invoice UDF's associated to the same order would share the same keyword: the *last* line to be sequentially parsed would indeed be the one which data will be recorded for future use (over-riding the values previously parsed/recorded for this Keyword).
2. At some stage, either inside the above "For each Order" loop or at a *later point of time*, the factory will have to check if all expected Invoice UDF Keywords were found in the Order Header. In case 1 or more Keywords were not found, the case would have to be escalated to HP (via the Prism Ticket process), since it will not be possible to execute all intended services on the units associated to this order.

4.4. Checking the validity of the Invoice UDF data values

In all cases, HP will be responsible of the data quality at the source (Order Entry). As a consequence, it shall normally not be needed to implement at *factory* level a check mechanism, to verify whether or not the data values obtained through EDI850 are supported by the service(s) they belong to.

However, in some cases, it will be decided to still implement such a mechanism, in order to avoid dropping any potential invalid order (from a strict EDI850 Invoice UDF data value standpoint) to the production floor – and discover really late that the order could in fact *not* be fulfilled.

4.4.1. Non-UPS-enabled factories

The MATINFO3 ("CNRS") deal is definitely an exception here, and factories are requested to continue performing a data validity check on the "2-letter entity identifier" once they will have switched their OSS/FTP based capability to EDI850.

In this particular case, the check is actually not here to detect *typos* at Order Entry level (because the "Service" Invoice UDF is automatically generated based upon the User login information), but to ensure that an *unsupported* entity (in the scope of the *deal* – but *not* in the scope -yet- of the related Configuration Services) which would place an order by mistake could be detected before the related units are assembled on the production line (and gets cancelled). The idea is also to detect cases of *missing* entity in EDI850.

On the next page you will find how the Escalation Process for MATINFO3 deal looks like once the source for Entity becomes EDI850.

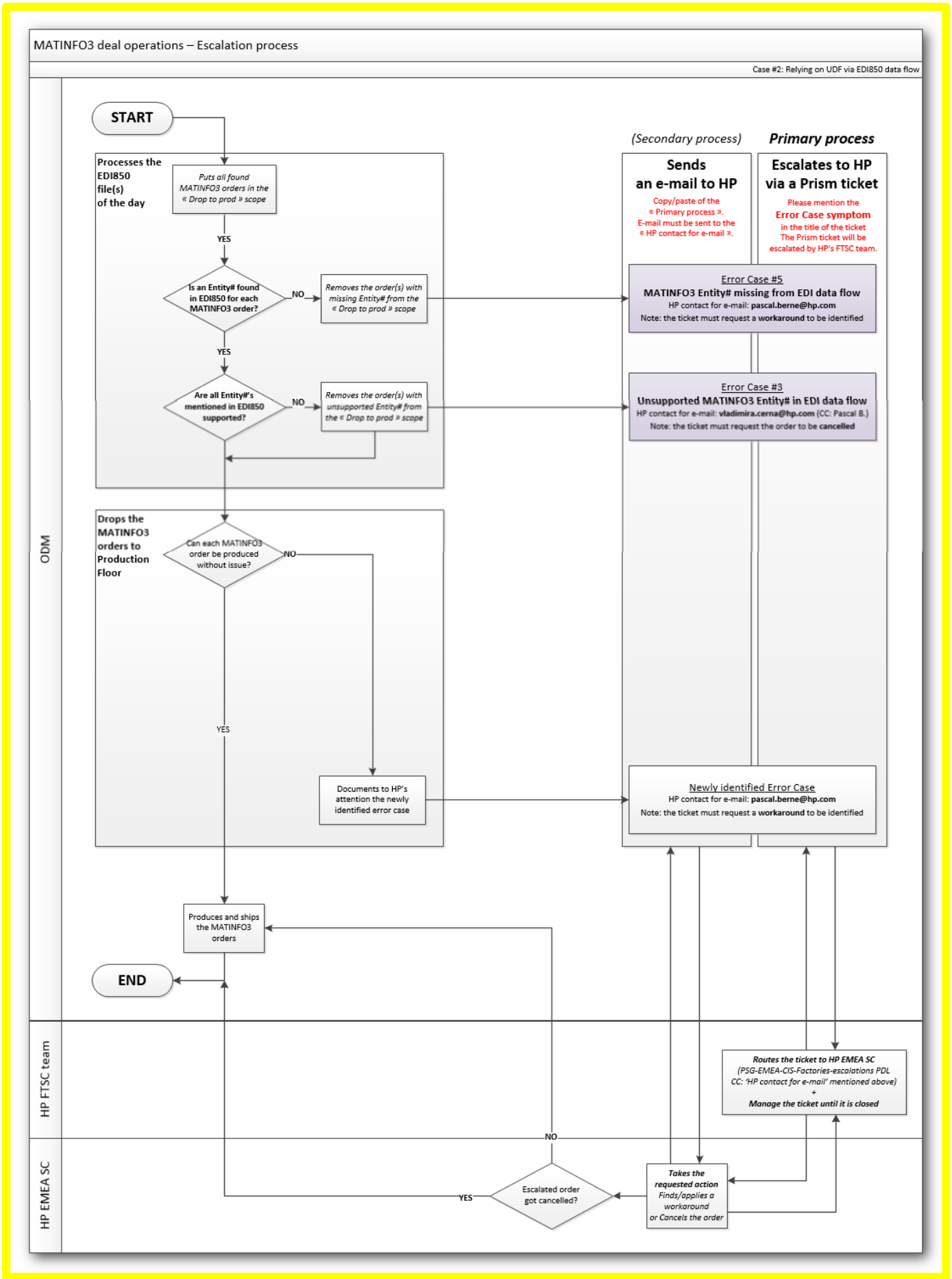
It can be compared to the Escalation Process which was provided in the Chapter 3.7.6. of the "HP CNRS on Notebooks requirements to ODM's" specification (HP Part# 468514-100).

4.4.1. UPS-enabled factories

UPS as a tool does not contain any data validity check mechanism such as the one we would be looking for with Invoice UDF data values.

Until the HP team defines how such checks (if any) could be best managed, no data validity check mechanism would be implemented.

The process documented on the next page does therefore *not* apply to UPS-enabled factories.


MATINFO3 deal operations – Escalation process
Case #2: Relying on UDF via EDI850 data flow
ODM
HP FTSC team
HP EMEA SC
START
Processes the EDI850 file(s) of the day
Puts all found MATINFO3 orders in the « Drop to prod » scope
YES
Is an Entity# found in EDI850 for each MATINFO3 order?
NO
Removes the order(s) with missing Entity# from the « Drop to prod » scope
YES
Are all Entity#'s mentioned in EDI850 supported?
NO
Removes the order(s) with unsupported Entity# from the « Drop to prod » scope
Drops the MATINFO3 orders to Production Floor
Can each MATINFO3 order be produced without issue?
NO
YES
Documents to HP's attention the newly identified error case
Produces and ships the MATINFO3 orders
END
(Secondary process)
Primary process
Sends an e-mail to HP
Copy/paste of the « Primary process ». E-mail must be sent to the « HP contact for e-mail ».
Escalates to HP via a Prism ticket
Please mention the Error Case symptom in the title of the ticket
The Prism ticket will be escalated by HP's FTSC team.
Error Case #5
MATINFO3 Entity# missing from EDI data flow
HP contact for e-mail: pascal.berne@hp.com
Note: the ticket must request a workaround to be identified
Error Case #3
Unsupported MATINFO3 Entity# in EDI data flow
HP contact for e-mail: vladimira.cerna@hp.com (CC: Pascal B.)
Note: the ticket must request the order to be cancelled
Newly identified Error Case
HP contact for e-mail: pascal.berne@hp.com
Note: the ticket must request a workaround to be identified
Routes the ticket to HP EMEA SC
(PSG-EMEA-CIS-Factories-escalations PDL
CC: 'HP contact for e-mail' mentioned above)
+
Manage the ticket until it is closed
NO
Escalated order got cancelled?
YES
Takes the requested action
Finds/applies a workaround
or Cancels the order

5. Pilot Plan for the associated capability

5.1. Pilot Contents

Before the new capability described in this document is moved to production at factory level, HP will perform an audit of the local capability.

In order for this audit to take place, one or more **Pilot Projects** will be managed in the Everest tool, with one or more **Pilot Services** inside.

Once the setup of the aforementioned Projects will have been completed, one or more **Pilot Orders** will be created in front of each Pilot Project.

Different services will be planned, depending on whether or not the targeted factory has got the UPS tool locally released.

5.2. Pilot Timeline

The timeline of the Pilots will be defined on a case-by-case basis with each targeted factory.

Factories getting orders from MATINF03 (“CNRS”) deal will be expected to prioritize the implementation of their “EDI850 UDI usage for Configuration Services” capability, in order to replace without delay the “Phase 1” OSS/FTP based solution currently in place for the fulfillment of MATINF03 orders with “dynamic” (multi-entity) Configuration Services.

6. Examples

6.1. Short structured UDI lines (Invoice UDF's) – Valide Case

Native UDI (compilation of real life examples - with end of lines represented as ■)

```
Service:AN■
UDF:MGB-NR: 292419;GLN-4306286002902■
Division name::PROCUREMENT■
Local installation contact phone number:Ouahib.Moudjahid@med.ge.com■
Royal London cost centre number:i224■
Laskutusviitteet:BR202089 TYKS 15T0014■
Phone number:+43 6245 794 365■
Server support level - :7d x 24 h x 6 h Call to repair■
Seller-Id @ST-Micro:V0HP010■
Please choose refresh/SWAP/buffer/ad hoc :buffer■
```

Resulting data in EDI850 file

```
N9~ZZ~User Defined Information
MSG~Service:AN
MSG~UDF:MGB-NR: 292419;GLN-4306286002902
MSG~Division name::PROCUREMENT
MSG~Local installation contact phone number:Ouahib.Moudjahid@med.ge.com
MSG~Royal London cost centre number:i224
MSG~Laskutusviitteet:BR202089 TYKS 15T0014
MSG~Phone number:+43 6245 794 365
MSG~Server support level - :7d x 24 h x 6 h Call to repair
MSG~Seller-Id @ST-Micro:V0HP010
MSG~Please choose refresh/SWAP/buffer/ad hoc :buffer
```

Parsed UDI (assuming that all Invoice UDF's are used by Configuration Services)

Invoice UDF #1	UDF keyword:	`Service`
	UDF separator:	`:`
	UDF value:	`AN`
Invoice UDF #2	UDF keyword:	`UDF`
	UDF separator:	`:`
	UDF value:	`MGB-NR: 292419;GLN-4306286002902`
Invoice UDF #3	UDF keyword:	`Division name`
	UDF separator:	`::`
	UDF value:	`PROCUREMENT`
Invoice UDF #4	UDF keyword:	`Local installation contact phone number`
	UDF separator:	`:`
	UDF value:	`Ouahib.Moudjahid@med.ge.com`
Invoice UDF #5	UDF keyword:	`Royal London cost centre number`
	UDF separator:	`:`
	UDF value:	`i224`

(...to be continued...)

(...continued...)

Invoice UDF #6	UDF keyword:	`Laskutusviitteet`
	UDF separator:	`:`
	UDF value:	`BR202089 TYKS 15T0014`
Invoice UDF #7	UDF keyword:	`Phone number`
	UDF separator:	`:`
	UDF value:	`+43 6245 794 365`
Invoice UDF #8	UDF keyword:	`Server support level -`
	UDF separator:	`:`
	UDF value:	`7d x 24 h x 6 h Call to repair`
Invoice UDF #9	UDF keyword:	`Seller-Id @ST-Micro`
	UDF separator:	`:`
	UDF value:	`V0HP010`
Invoice UDF #10	UDF keyword:	`Please choose refresh/SWAP/buffer/ad hoc`
	UDF separator:	`:`
	UDF value:	`buffer`

6.2. Several unstructured UDI lines – Valid case

Native UDI (wrapped for legibility purpose, with end of lines represented as ■)

```
GOODS IN ZONE5■
WESTWARD HOUSE■
15-17 ST JAMES STREET■
PAISLEY■
PA3 2HL■
```

Resulting data in EDI850 file

```
N9~ZZ~User Defined Information
MSG~GOODS IN ZONE5
MSG~WESTWARD HOUSE
MSG~15-17 ST JAMES STREET
MSG~PAISLEY
MSG~PA3 2HL
```

Parsed UDI – *Not Applicable (no Invoice UDF)*

6.3. 1 unstructured and 1 structured line – Valid case

Native UDI (wrapped for legibility purpose, with end of lines represented as ■)

```
This is the first Native UDI line, and it is UNSTRUCTURED.■
On Purpose: This is the second Native UDI line, and it is STRUCTURED.■
```

Resulting data in EDI850 file

```
N9~ZZ~User Defined Information
MSG~This is the first Native UDI line, and it is UNSTRUCTURED.
MSG~On Purpose: This is the second Native UDI line, and it is STRUCTURED.
```

Parsed UDI (assuming that the Invoice UDF is used by Configuration Services)

Invoice UDF #1	UDF keyword:	`On Purpose`
	UDF separator:	`:`
	UDF value:	`This is the second Native UDI line, and it is STRUCTURED.`

6.4. 1 unstructured and 1 structured line – Invalid case

Native UDI (wrapped for legibility purpose, with end of lines represented as ■)

```
This is the first Native UDI line, and it is UNSTRUCTURED.■
On Purpose: This is the second Native UDI line, and it is intentionally STRUCTURED.■
```

Resulting data in EDI850 file

```
N9~ZZ~User Defined Information
MSG~This is the first Native UDI line, and it is UNSTRUCTURED.
MSG~On Purpose: This is the second Native UDI line, and it is intentionall
MSG~y STRUCTURED.
```

Parsed UDI (assuming that the Invoice UDF is used by Configuration Services)

Invoice UDF #1	UDF keyword:	`On Purpose`
	UDF separator:	`:`
	UDF value:	`This is the second Native UDI line, and it is intentionall`

Note that the UDF value is *truncated*, since an Invoice UDF used by Configuration Services isn't supposed to be longer than 70 characters. This case should therefore never occur in real life…

6.5. 1 unstructured line being interpreted as being structured

Native UDI (wrapped for legibility purpose, with end of lines represented as ■)

```
This is the first Native UDI line, and it is UNSTRUCTURED.■
Warning: This is the 2nd line, and it is meant to be UNSTRUCTURED.■
```

Resulting data in EDI850 file

```
N9~ZZ~User Defined Information
MSG~This is the first Native UDI line, and it is UNSTRUCTURED.
MSG~Warning: This is the 2nd line, and it is meant to be UNSTRUCTURED.
```

Parsed UDI (though it is NOT a real Invoice UDF)

Invoice UDF #1	UDF keyword:	`Warning`
	UDF separator:	`:`
	UDF value:	`This is the 2nd line, and it is meant to be UNSTRUCTURED.`

Note that the second UDI line is detected as being an Invoice UDF, while in fact it was not *intentionally* structured by the User. There is no risk though that it generates issues during operations.

7. Document maintenance / Feedbacks

As a reader of this document, you are highly welcome to participate to enhancements to it, by providing your feedbacks to Pascal Berne (pascal.berne@hp.com) from HP CIS EMEA team.

Revision history

v1.0	February 4, 2014	Original revision
v1.1	February 28, 2014	*Page #8:* Added 2 new cases of “forbidden characters”. *Page #11:* Clarified how a UPS service will document the special case of Invoice UDF data values that the service depends on. *Page #13:* Fixed a typo in the Regular Expressions to parse the Keyword and the Separator in an Invoice UDF line. *Pages #14 through #16:* Added a new Chapter to cover the Invoice UDF data value validity check question.

Hewlett-Packard Company

3000 Hanover Street

Palo Alto, CA 94304

www.hp.com