ECHA
EUROPEAN CHEMICALS AGENCY

UFI Developers Manual

March 2019

Disclaimer

This document aims to assist users in complying with their obligations under the CLP Regulation. However, users are reminded that the text of the CLP Regulation is the only authentic legal reference and that the information in this document does not constitute legal advice. Usage of the information remains under the sole responsibility of the user. The European Chemicals Agency does not accept any liability with regard to the use that may be made of the information contained in this document.

Version	Changes	
1.2	Clarification / Update of Chapter 2.1.3 "Step 3 – Character reorganisation".	11/03/2019
1.1	Converted Commission document to an ECHA publication: ECHA visual identity applied, ECHA disclaimer added and list of participating countries updated	2/07/2018
1.0	Same as version 0.3 – Marked 'Final'	04/10/2016
0.3	Corrected typos, minor enhancements	16/09/2016
0.2	Corrected typos, integrated comments from ECHA and DG GROW	08/08/2016
0.1	First complete draft version	14/07/2016

UFI Developers Manual

Reference: ECHA-19-H-05-EN
ISBN: 978-92-9481-116-5
Cat. Number: ED-02-19-147-EN-N
DOI: 10.2823/24159
Publ.date: March 2019
Language: EN

© European Chemicals Agency, 2019
Cover page © European Chemicals Agency

If you have questions or comments in relation to this document please send them (quote the reference and issue date) using the information request form. The information request form can be accessed via the Contact ECHA page at:
http://echa.europa.eu/contact

European Chemicals Agency

Mailing address: P.O. Box 400, FI-00121 Helsinki, Finland
Visiting address: Annankatu 18, Helsinki, Finland

Table of Contents

Tables

Figures

1. Introduction

This document describes the technical means available to developers of software solutions to create and validate the Unique Formula Identifiers (UFIs) that have to be printed or affixed on labels of hazardous mixtures in accordance with the Annex related to Article 45 of [CLP REGULATION] (see references in section 1.3; the Annex is a standalone document referred to as [CLP ANNEX]).

Section 1.1 of this document reminds the **purpose of the UFI** and details the **base input information** that will be needed to create one. However it does not discuss the situations where a UFI must be created for a mixture or when such a UFI needs to be updated. The reader who needs to handle these aspects is advised to consult the [CLP ANNEX] and the guidance material available from the section of ECHA's website dedicated to the submission of information to Poison Centres (referred to as [GUIDANCE]):

https://poisoncentres.echa.europa.eu/

The rest of the document is composed of the technical information needed to build a practical software implementation creating and validating UFIs.

- **Chapter 2** gives the **algorithm** that can applied to create a UFI and explains how an existing UFI can be validated with respect to a series of criteria.
- Chapter 3 details the public Application Programming Interface (API) offered by the "UFI Generator" application run by ECHA.

  This API, available over **REST and SOAP web services**, readily implements the algorithm of chapter 2; it can then be used by any system where calling an existing web service is a better solution than implementing the algorithm of chapter 2 from scratch.

  Note that the UFI Generator also features a web interface with similar functionalities for human users. This interface is available from the aforementioned link to ECHA's Poison Centres web site.

The pros and cons of approaches in chapter 2 and chapter 3 (that is, implementing the algorithm or calling web services) are not detailed as they largely depend on the capabilities of the system that will generate the UFIs, the resources available for the development, etc. One can however note that an implementation of the algorithm specified in chapter 2 will most likely yield better performances and that it does not rely on remote resources. It will thus be immune to the unavailability of the network infrastructure and possible downtime of the UFI Generator.

The document does not expect a particular programming language to be used, such as Java or .NET. The discussion is kept a sufficiently high level to allow using any language.

Definitions of abbreviations used in this document are available in section 1.4.

1.1 The Unique Formula Identifier

The Unique Formula Identifier (UFI) establishes an unambiguous link between a mixture and the information submitted to Poison Centres about that mixture. It complements the other means used by Poison Centres to identify the source of poisoning as basis for clinical toxicological risk assessment and to propose the right medical treatment. E.g. the UFI will be used to distinguish two formulations sold under the same trade name. [CLP ANNEX] therefore requires to print or affix a UFI on the label of a product and mention the UFI in the submission of information to Poison Centres.

The UFI shall be unique. Therefore, the same UFI can never be assigned to two different mixtures[1]. Practically, this is achieved by constructing UFIs from two parameters:

- A VAT number (VATIN), as used in countries of the EEA;
- A numerical formulation number between 0 and 268.435.455 (included).

Assuming a proper use and management of these by companies, e.g. the same formulation number shall not be used for different mixtures under the same VATIN, the uniqueness of the UFI is guaranteed. The [GUIDANCE] can be consulted for recommendations about correct uses of VATIN and formulation numbers (e.g. what VATIN is suitable when there are multiple companies in the supply chain for a product).

Companies without a VAT number

In some countries, under very specific conditions, companies are not required to have a VATIN. These companies can nonetheless create UFIs like any other company with a company identification number; for them the VATIN is replaced in the algorithm of chapter 2 by a "company key".

A company key can be obtained from the tab "Get a company key" of the UFI Generator on ECHA's Poison Centres web site (see link above and [UI MANUAL]). It is advised for companies without a VATIN that want to implement the algorithm of chapter 2 to issue one company key and store it for any future use. I.e. a new company key will not be issued for each individual UFI; companies should also not attempt to automate the procurement of company keys.

Note that a company key is not necessary if only the web services of chapter 3 are used. With these web services the VATIN parameter can be omitted from the requests to create UFIs; the UFI Generator (that provides the web services) will automatically fill in the missing information while ensuring that the uniqueness of the UFI is respected.

1.2 Conventions

The reader should note the following before continuing reading the next chapters:

- This document generally distinguishes the two parts constituting a VATIN:
  - The code of the country issuing the VATIN and designated as "VAT country code".

    Uppercase ISO-3166-1 alpha-2 country codes are used rather than the prefix found in the VATIN itself. The codes GR (rather than EL) is thus used for Greece. The admissible codes are those listed in Table 2-1.
  - The national specific part (following the prefix) and designated as "VAT number". E.g. "U12345678" in the VATIN "ATU12345678".

    Note that, as shown in this example, for some countries the "number" can include other characters than digits.
- The formulation number is necessarily a numerical value.

  Formulation codes used internally by companies, whether they contain non numerical characters or are outside the range [0, 268.435.455], need to be converted to a suitable numerical value prior to being used in the UFI creation algorithm or used as a

[1] In the sense defined by [CLP ANNEX]; see notably the latter and the [GUIDANCE] for a discussion of grouped submission of variants of a mixture or mixtures differing by the concentration of some components within well-defined ranges.

parameter of web service operations. Conversion rules are outside the scope of this document and can be freely defined by companies or third-party software vendors.

1.3 References

Documents referred to in this manual are listed below.

Table 1-1: External references

Reference	Description
[CLP REGULATION]	Regulation (EC) No 1272/2008 of the European Parliament and of the Council of 16 December 2008 on classification, labelling and packaging of substances and mixtures, amending and repealing Directives 67/548/EEC and 1999/45, and amending Regulation (EC) No 1907/2006 O. J. L 353, 31.12.2008 http://eur- lex.europa.eu/search.html?DTN=1272&DTA=2008&qid=1451989819998&CASE_LAW_SUMMARY=false&DTS_DOM=ALL&excConsLeg=true&type=advanced&SUBDOM_INIT=ALL_ALL&DTS_SUBDOM=ALL_ALL Latest consolidated version in English: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX%3A02008R1272-20150601&qid=1449585466596&from=EN[2]
[CLP ANNEX]	ANNEX to the Commission Regulation amending Regulation (EC) No 1272/2008 of the European Parliament and of the Council on classification, labelling and packaging of substances and mixtures by adding a new Annex on harmonised information relating to emergency health response
[GUIDANCE]	Guidance (to be written) https://poisoncentres.echa.europa.eu/
[UI MANUAL]	UFI Generator application User Guide

[2] Link to the latest, as per January 2016, consolidated version in English, compiling amendments M1 to M9 and corrigendum C1 and C2.

1.4 Abbreviations

Abbreviations used in this manual are listed below in alphabetic order.

Table 2-2: Abbreviations

Acronym	Definition
API	Application Programming Interface[3]
CLP	Classification, Labelling and Packaging
CSV	Comma Separated Values[4]
EEA	European Economic Area
MS	Member State
PC	Poison Centre
REST	Representational State Transfer[5]
SOAP	Simple Object Access Protocol[6]
UFI	Unique Formula Identifier
VAT	Value-Added Tax
VATIN	VAT Identification Number
WSDL	Web Services Description Language[7]
XML	Extensible Mark-up Language[8]

[3] https://en.wikipedia.org/wiki/Application_programming_interface
[4] https://en.wikipedia.org/wiki/Comma-separated_values
[5] https://en.wikipedia.org/wiki/Representational_state_transfer
[6] https://en.wikipedia.org/wiki/SOAP
[7] https://en.wikipedia.org/wiki/Web_Services_Description_Language
[8] https://en.wikipedia.org/wiki/XML

2. UFI algorithm

This chapter gives the algorithm that applies to create a UFI and explains how an existing UFI can be validated with respect to a series of criteria.

Examples of UFI creation and validation following the algorithm are shown in Annex A.

Samples of UFIs for various VATIN (or company key) and formulation numbers are listed in Annex B. These can be used to validate if the algorithm is properly implemented (if necessary the *validateUFI* operation of the web services in chapter 3 can also be used for that purpose).

2.1 Creating a UFI

Below we summarise and clearly define the steps for the construction of a UFI from two input data (see introduction on UFI in section 1.1 for motivation):

- A VATIN or a company key for companies without a VATIN;
- A formulation number.

Creating a UFI is a 4 step process explained in the rest of this section:

1. A numerical "payload" value is computed based on the input information.

2. The payload numerical value is written in a character set excluding ambiguous characters.

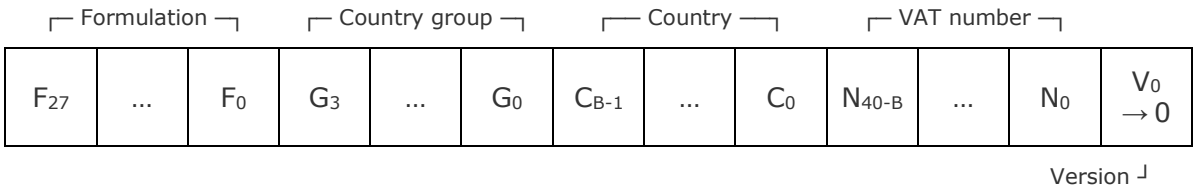
3. The characters are re-organised to spread information from the formulation number and VATIN across the whole UFI.

   This step is meant to enable filtering alternatively on both the formulation number and VATIN when incremental searches[9] are implemented to retrieve information from a database of UFI.

4. A checksum is calculated and prepended to the result to give the complete UFI

2.1.1 Step 1 – UFI payload numerical value

The UFI payload numerical value is constructed in binary as shown in the following diagram (where cells represent the bits composing that value),



Where

- F_{27} to F_0 are 28 bits encoding the input formulation number.

[9] That is, searches displaying the results on-the-fly, while characters are being keyed in (https://en.wikipedia.org/wiki/Incremental_search).

- G_3 to G_0 are 4 bits encoding the country group given by Table 2-1 for the input VAT country code.

  Since not all VAT numbers require the same number of bits (e.g. numbers in Lichtenstein are much smaller than those in France) the usage of the 41 following bits labelled *C* and *N* (country and VAT number in the diagram) change depending on this country group to make better use of the "space" available in the UFI payload.

  When the company does not have a VATIN the 4 bits of the country group G must be set to 0.

- C_{B-1} to C_0 are *B* bits encoding the VAT country code inside the group determined by *G*. The value of *B* is given by Table 2-1.

  The number of bits reserved for the country code depends on the country group. For some groups *B* = 0 in Table 2-1, meaning that there is no explicit country code[10].

  When the company does not have a VATIN there is no *C* (similarly to cases with VATIN where *B* = 0).

- N_{40-B} to N_0 are 41 – *B* bits encoding the VAT number.

  The number of bits reserved for the VAT number thus also depends on the country group. 41 bits is the maximum number of bits that can be used to store the VAT number; it is attained for country groups where *B* = 0 (without explicit VAT country code).

  For VAT numbers including non-numerical characters a numerical equivalent must be calculated prior to storing it in *N*. The calculation rules are given on a per-VAT country code basis in Table 2-2 (the table also reminds the list of countries for which the VAT number can be used as-is).

  When the company does not have a VATIN the 41 bits are used to encode the company key (similarly to cases with VATIN where *B* = 0; see note on company key issuance in section 1.1).

- V_0 is a (single) bit encoding the UFI version and must be set to 0.

The UFI payload numerical value is thus a 74-bit number with 28 bits dedicated to the formulation number, 45 to the VATIN (or company key) and 1 to the version identification.

Table 2-1: Country groups and codes lookup table

Country		Country group code (G)	Number of bits for country code (B)	Country code (C)
Group code 0 is reserved for company keys		0	None	None
FR	France	1		
GB	United Kingdom	2		
LT	Lithuania	3	1	0
SE	Sweden			1

[10] In these cases there is actually only one country in the group.

HR	Croatia	4	4	0
IT	Italy			1
LV	Latvia			2
NL	The Netherlands			3
BG	Bulgaria	5	7	0
CZ	Czech Rep.			1
IE	Ireland			2
ES	Spain			3
PL	Poland			4
RO	Romania			5
SK	Slovakia			6
CY	Cyprus			7
IS	Iceland			8
BE	Belgium			9
DE	Germany			10
EE	Estonia			11
GR	Greece			12
NO	Norway			13
PT	Portugal			14
AT	Austria			15
DK	Denmark			16
FI	Finland			17
HU	Hungary			18
LU	Luxemburg			19
MT	Malta			20
SI	Slovenia			21
LI	Lichtenstein			22

Table 2-2: Rules for VAT number conversion to numerical value

VAT country		VAT number pattern	Conversion to numerical value (denoted V)
AT	Austria	(AT)U[0-9]{8}	The VAT number is only composed of a numerical part, V is trivially equal to that numerical part. Notes: • For countries with a numerical part of varying length (BG, CZ, LT and RO), we assume that e.g. RO012 or RO0012 are effectively the same numbers as RO12. That is, the varying length does not require any specific handling. • Some VAT numbers have fixed letters in well-defined places (e.g. all numbers in Austria start with "U"). These fixed letters are omitted from the conversion.
BE	Belgium	(BE)0[0-9]{9}	
BG	Bulgaria	(BG)[0-9]{9,10}	
CZ	Czech Republic	(CZ)[0-9]{8,10}	
DE	Germany	(DE)[0-9]{9}	
DK	Denmark	(DK)[0-9]{8}	
EE	Estonia	(EE)[0-9]{9}	
GR	Greece	(EL|GR)[0-9]{9}	
FI	Finland	(FI)[0-9]{8}	
HR	Croatia	(HR)[0-9]{11}	
HU	Hungary	(HU)[0-9]{8}	
IT	Italy	(IT)[0-9]{11}	
LI	Liechtenstein	(LI)[0-9]{5}	
LT	Lithuania	(LT)([0-9]{9}|[0-9]{12})	
LU	Luxembourg	(LU)[0-9]{8}	
LV	Latvia	(LV)[0-9]{11}	

VAT country		VAT number pattern	Conversion to numerical value (denoted V)
MT	Malta	(MT)[0-9]{8}	
NL	The Netherlands	(NL)[0-9]{9}B[0-9]{2}	
NO	Norway	(NO)[0-9]{9}	
PL	Poland	(PL)[0-9]{10}	
PT	Portugal	(PT)[0-9]{9}	
RO	Romania	(RO)[0-9]{2,10}	
SE	Sweden	(SE)[0-9]{12}	
SI	Slovenia	(SI)[0-9]{8}	
SK	Slovakia	(SK)[0-9]{10}	
CY	Cyprus	(CY)[0-9]{8}[A-Z]	V = l . 10^8 + d, where • d is the numerical part (8 digits) • l is the value of the letter with A → 0, B → 1, … Z → 25
ES	Spain	(ES)[0-9A-Z][0-9]{7}[0-9A-Z]	V = (36 . c1 + c_2) . 10^7 + d, where • d is the numerical part (7 digits) • c_1 and are c_2 respectively the values of the first and last characters with 0 → 0, 1 → 1, … 9 → 9, A → 10, B → 11, … Z → 35
FR	France	(FR)[0-9A-Z]{2}[0-9]{9}	V = (36 . c_1 + c_2) . 10^9 + d, where • d is the numerical part (9 digits) • c_1 and are c_2 respectively the values of the first and second characters with 0 → 0, 1 → 1, … 9 → 9, A → 10, B → 11, … Z → 35

VAT country		VAT number pattern	Conversion to numerical value (denoted V)
GB	United Kingdom	(GB)([0-9]{9}([0-9]{3})?|[A- Z]{2}[0-9]{3})	The pattern allows for three types of numbers, V is calculated depending on the case as, The pattern allows for three types of numbers, V is calculated depending on the case as, • For 9-digit and 12-digit numbers, denoted d, $V = 2^{40} + d$ That is, the leftmost bit of the 41 bits reserved for the VAT number in the UFI payload numerical value is set to 1 to avoid collisions with the numbers in the point below; the 9 or 12-digit number is placed in the 40 remaining bits. • For 2-letter and 3-digit, $V = (26 . l_1 + l_2) . 10^3 + d$, where ○ d is the numerical part (3 digits) ○ l_1 and l_2 are respectively the values of the first and second letters with A → 0, B → 1, … Z → 25

VAT country		VAT number pattern	Conversion to numerical value (denoted V)
IE	Ireland	(IE)([0-9][A-Z*+][0-9]{5}[A-Z]|[0-9]{7}([A-Z]W?|[A-Z]{2}))	The pattern allows for three types of numbers, V is calculated depending on the case as, • For numbers matching [0-9][A-Z*+][0-9]{5}[A-Z] $V = (26 \cdot c_1 + c_2) \cdot 10^6 + d$, where ○ d is the numerical part (6 digits) composed of the concatenated digits in the pattern (i.e. [0-9] … [0-9]{5} …) ○ c_1 and are c_2 respectively the values of the first and last characters with A → 0, B → 1, … Z → 25, + → 26, * → 27 • For numbers matching [0-9]{7}[A-Z]W? or [0-9]{7}[A-Z]{2} $V = 2^{33} + ((26 \cdot c_2 + c_1) \cdot 10^7 + d)$, where ○ d is the numerical part (7 digits) ○ c_1 and are c_2 respectively the values of the 8_{th} and 9_{th} characters with A → 0, B → 1, … Z → 25; when the 9^{th} character is omitted $c_2 = 0$ (notice that c_2 is multiplied by 26 above, not c_1, unlike the other pattern in the point above)
IS	Iceland	(IS)[A-Z0-9]{6}	$V = 36^5 \cdot c_6 + 36^4 \cdot c_5 + \ldots + 36 \cdot c_2 + c_1$, where • c_i is the value of the i^{th} character, from right to left, with 0 → 0, 1 → 1, … 9 → 9, A → 10, B → 11, … Z → 35

2.1.2 Step 2 – UFI payload in base-31

The payload value obtained at step 1 is written[11] in base-31 using the representation of digits in Table 2-3. If necessary, the result is left-padded with 0 to obtain 15 characters.

Table 2-3: Base-31 character set

Digit	Repr.	Digit	Repr.	Digit	Repr.	Digit	Repr.	Digit	Repr.
0	0	7	7	14	F	21	P	28	W
1	1	8	8	15	G	22	Q	29	X
2	2	9	9	16	H	23	R	30	Y
3	3	10	A	17	J	24	S		
4	4	11	C	18	K	25	T		
5	5	12	D	19	M	26	U		
6	6	13	E	20	N	27	V		

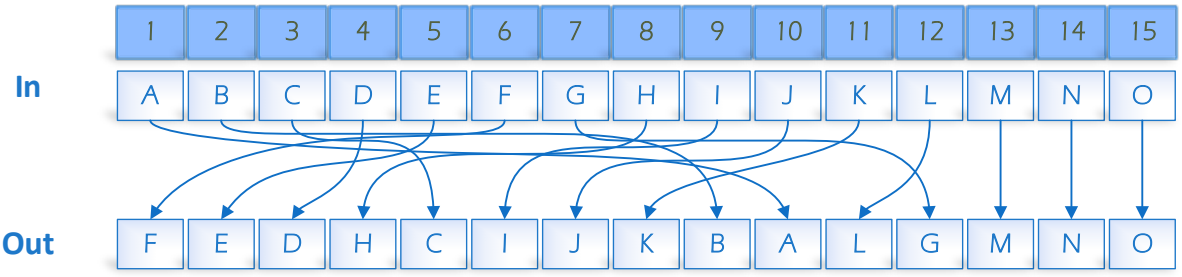
2.1.3 Step 3 – Character reorganisation

The 15 characters from step 2 are reorganised according to Table 2-4 which specifies the input string (in) and the output string (out); characters are numbered from 1 (left) to 15 (right). For example, the first character of the output string corresponds to the 6th character of the input string.

Table 2-4: UFI characters reorganisation tables

In	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Out	6	5	4	8	3	9	10	11	2	1	12	7	13	14	15

For example if we consider "A B C D E F G H I J K L M N O" as the **In** string then the **Out** string would be "F E D H C I J K B A L G M N O" as the following diagram depicts:



2.1.4 Step 4 – Checksum calculation

Finally the checksum character is prepended to the 15 characters from the previous step to give the 16-character UFI.

The checksum u_0 (that is, the leftmost character of the complete UFI) is calculated as

[11] Typically, by repeating a modulo 31 and an integer division by 31 until the remainder is 0.

follows and written using the representation of Table 2-3. Factors u_1 to u_{15} represent the 15 characters obtained after the reorganisation at step 3 (character 1 being at the left).

- $u_0 = (31 - (2 \,.\, u_1 + 3 \,.\, u_2 + \ldots + 16 \,.\, u_{15}) \text{ modulo } 31) \text{ modulo } 31$

2.2 Validating a UFI

When validating a UFI it is recommended to implement validations to detect the errors of Table 3-6; where technically

- VAL001 ensures that the UFI is composed of 16 characters (omitting the hyphens and possibly spaces, depending on the requirement of the particular client implementation);
- VAL002 ensures that the 16-character UFI it only composed of the characters in Table 2-3;
- VAL003 ensures that the UFI is consistent with respect to its checksum;

  The checksum can be validated by verifying if the expression below is equal to 0 (using the same notation for characters ui of the UFI as that of step 4 above).

  $(u_0 + 2 \,.\, u_1 + 3 \,.\, u_2 + \ldots + 16 \,.\, u_{15}) \text{ modulo } 31 \stackrel{?}{=} 0$
- VAL004 ensures the consistency of the group code and country encoding with respect to entries defined in Table 2-1;

  Note that the VAT number can also be verified with respect to national patterns; notably, the encoding of non-numerical numbers is bijective and can thus be inverted prior to applying a validation pattern[12].
- VAL005 ensures that the right-most bit for the version is 0.

[12] The UFI Generator does not implement such verifications, only the consistency group code and country encoding is verified.

3. UFI Generator web services

The UFI Generator offers SOAP 1.1 and REST web services with similar operations to create and validate UFI(s). A remote system can then call the web service that is the most suitable for its specific case.

The operations are listed in Table 3-1 and their interfaces are further described in sections 3.1 (for REST) and 3.2 (for SOAP). Section 3.3 lists the error codes that can be returned by the UFI creation and validation operations.

For both UFI creation operation the UFI Generator will enforce a limit on the count of formulation numbers in a single call to the operation; the limit is the same as that of the UI (see [UI MANUAL] for a figure).

Table 3-1: Web service operations

Operation	Input arguments	Output	Description
createUFIByCount	• VATIN (opt.) • Start formulation number • Count	List of UFIs	Create a succession of "count" UFIs starting with the given formulation number for the given VATIN. The latter can be omitted by companies without a VATIN.
createUFIByList	• VATIN (opt.) • List of formulation numbers	List of UFIs	Create UFIs for the given formulation numbers and VATIN. The latter can be omitted by companies without a VATIN.
validateUFI	• UFI	Boolean	Validate the correctness of a UFI with respect to a series of rules (see errors in Table 3-6).

3.1 REST web service

The REST flavoured operations of Table 3-1 are described in the sections below.

The base path to construct the URLs is: https://ufi.echa.europa.eu/ufi

3.1.1 createUFIByCount

Table 3-2: REST operation createUFIByCount

Method	GET	
URL	base path + /createUFIByCount	
URL parameters	vatCountryCode (optional)	VAT country code. This argument can be omitted, provided that *vatNumber* below is omitted too, to indicate that the company does not have a VATIN and that the UFI Generator must take ad-hoc measures when generating the UFI.

<table>
<tr><td></td><td>vatNumber
(optional)</td><td>VAT number.

Notes that this argument must be URL-encoded if it contains characters not allowed in URLs.

This argument can be omitted, provided that vatCountryCode above is omitted too, to indicate that the company does not have a VATIN.</td></tr>
<tr><td></td><td>startFormulation Number</td><td>First formulation number in the succession.</td></tr>
<tr><td></td><td>count</td><td>Count of UFIs to create with successive formulation numbers.</td></tr>
<tr><td>Request examples</td><td colspan="2">With VATIN
<code>createUFIByCount?
vatCountryCode=BE&vatNumber=0123456789&
startFormulationNumber=12&count=34</code>
Without VATIN
<code>createUFIByCount?
startFormulationNumber=12&count=34</code></td></tr>
<tr><td rowspan="4">Success response</td><td>HTTP code</td><td>200 (OK)</td></tr>
<tr><td>Format</td><td>JSON</td></tr>
<tr><td>Content example</td><td><code>[
{
"ufi":"511060T3400CSP6U",
"formulationNumber":{"value":12},
"formattedUfi":"5110-60T3-400C-
SP6U"
},
{
"ufi":"3410Q0GGE00VE0SW",
"formulationNumber":{"value":13},
"formattedUfi":"3410-Q0GG-E00V-
E0SW"
}
…
]</code></td></tr>
<tr><td>Notes</td><td>The client should not assume any particular order in the sequence of results. The client should thus always construct the association between UFI and formulation number from both elements in the result.</td></tr>
<tr><td rowspan="4">Error response</td><td>HTTP code</td><td>400 (bad request)</td></tr>
<tr><td>Format</td><td>JSON</td></tr>
<tr><td>Content example</td><td><code>{
"message":"Formulation number count
is invalid.",
"code":"UFI010"
}</code></td></tr>
<tr><td>Notes</td><td>The error codes are those in Table 3-5.

The error messages are always in English.</td></tr>
</table>

3.1.2 createUFIByList

Table 3-3: REST operation createUFIByLis

Method	POST with multipart/form-data	
URL	base path + /createUFIByList	
Content disposition names	vatCountryCode (optional)	Same as Table 3-2. Both parts can be omitted to indicate that the company does not have a VATIN.
	vatNumber (optional)	
	file	List of formulation numbers for which UFIs must be created. One formulation number per line. A filename content disposition parameter must be supplied; its value is irrelevant and can be left blank.

Request example	<pre>------=_Part_2_734057280.1467970322787
	
	Content-Disposition: form-data;
	name="vatCountryCode"
	
	BE
	------=_Part_2_734057280.1467970322787
	Content-Disposition: form-data; name="vatNumber"
	
	0123456789
	------=_Part_2_734057280.1467970322787
	Content-Disposition: form-data; name="file";
	filename=””
	
	12
	13
	------=_Part_2_734057280.1467970322787--</pre>	

Success response	Same as Table 3-2.	
Error response	Same as Table 3-2.	

3.1.3 validateUFI

Table 3-4: REST operation validateUFI

Method	GET	
URL	base path + /validateUFI	
URL parameters	ufi	UFI to validate. Note that the hyphens (-) or spaces are not taken into account for the validation.

Request examples	Valid UFI
	
	`validateUFI?`
	`ufi=5110-60T3-400C-SP6U`
	
	Invalid UFI (with dot instead of dash)
	
	`ufi=5110%2E60T3-400C-SP6U`	

Success response	HTTP code	200 (OK)
	Format	JSON

	Content example, valid UFI	`{`
		`"message":"",`
		`"code":"", "valid":true`
		`}`

	Content example, invalid UFI	`{`
		`"message":"This UFI is not valid: It does not contain 16 characters.", "code":"VAL001", "valid":false`
		`}`

	Notes	The error codes are those in Table 3-6. The error messages are always in English.
Error response	None (validating an invalid UFI returns a success response with a valid flag set to false; see example above)	

3.2 SOAP web service

The SOAP 1.1 web service follows a document/literal style binding. A WSDL describing the service and the port where it can be contacted can be found at:

https://ufi.echa.europa.eu/ufi/ws/eu/echa/ufi/ws/wsdl/UFIGenerator.wsd

Examples of each operation of Table 3-1 are given in the sections below to clarify how SOAP requests, responses and faults are constructed.

3.2.1 Requests for createUFIByCount and createUFIByList

Notes:

- For the two operations the "vatin" element can be omitted (i.e. its definition has a minOccurs="0") to indicate that the company does not have a VATIN and that the UFI Generator must take ad-hoc measures when generating the UFI.
- For the createUFIByList the element "formulationNumber" can be repeated if UFIs must be generated for different formulation numbers (i.e. its definition has a maxOccurs="unbounded").

Figure 3-1: SOAP request for createUFIByCount

```
<createUFIByCountRequest
xmlns="urn:eu:europa:ec:grow:pc:ufi:generator:types:1">
    <vatin>
        <countryCode>BE</countryCode>
        <number>0123456789</number>
    </vatin>
    <startFormulationNumber>12</startFormulationNumber>
    <count>34</urn:count>
</createUFIByCountRequest>
```

Figure 3-2: SOAP request for createUFIByList

```
<createUFIByListRequest
xmlns="urn:eu:europa:ec:grow:pc:ufi:generator:types:1">
    <vatin>
        <countryCode>BE</countryCode>
        <number>0123456789</number>
    </vatin>
    <formulationNumber>12</formulationNumber>
    <formulationNumber>13</formulationNumber>
    ...
</createUFIByListRequest>
```

3.2.2 Response to createUFIByCount and createUFIByList

Note that each UFI is accompanied by its corresponding formulation number. The client should not assume any particular order in the sequence of results; notably, the UFIs might not be sequential for createUFIByCount or match the order in the request for createUFIByList. The client should thus always construct the association between UFI and formulation number from both elements in the result.

Figure 3-3: SOAP response to createUFIByCount and createUFIByList

```
<createUFIResponse
xmlns="urn:eu:europa:ec:grow:pc:ufi:generator:types:1">
    <result>
        <ufi>5110-60T3-400C-SP6U</ufi>
        <formulationNumber>12</formulationNumber>
    </result>
    <result>
        <ufi>3410-Q0GG-E00V-E0SW</ufi>
        <formulationNumber>13</formulationNumber>
    </result>
    ...
</createUFIResponse>
```

3.2.3 Fault for createUFIByCount and createUFIByList

Notes:

- In case of error caused by the client the "detail" element contains an "invalidArgumentFault" element describing why the arguments are considered invalid by referring to the errors in Table 3-5 (the message is always in English).
- Multiple "error" elements can possibly be given in the "invalidArgumentFault" element (i.e. its definition has a maxOccurs="unbounded"). The client should however not assume that, if multiple errors exist in the request, they will all be returned at once.
- More generally, the client should be ready handle any SOAP fault (e.g. also faults with code SOAP-ENV:Server).

Figure 3-4: SOAP fault for createUFIByCount and createUFIByList

```
<SOAP-ENV:Fault xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <faultcode>SOAP-ENV:Client</faultcode>
    <faultstring xml:lang="en">Invalid
argument</faultstring>
    <detail>
        <invalidArgumentFault
xmlns="urn:eu:europa:ec:grow:pc:ufi:generator:types:1">
          <error>
              <errorCode>UFI001</errorCode>
              <message>Invalid or out of bound formulation
number</message>
          </error>
        </invalidArgumentFault>
    </detail>
</SOAP-ENV:Fault>
```

3.2.4 Request for validateUFI

Note that the hyphens (-) or spaces are not taken into account for the validation.

Figure 3-5: SOAP request for validateUFI

```
<validateUFIRequest
xmlns="urn:eu:europa:ec:grow:pc:ufi:generator:types:1">
    <ufi>5110-60T3-400C-SP6U</ufi>
</validateUFIRequest>
```

3.2.5 Response to validateUFI

Notes:

- When "valid" is false an "error" element referring to the errors in Table 3- 6 indicates why the UFI is invalid (the message is always in English).
- A SOAP fault is not returned when calling validateUFI with an invalid UFI as it is considered a legit use of the operation. The client should still be ready handle SOAP faults (e.g. faults with code SOAP-ENV:Server).

Figure 3-6: SOAP response to validateUFI for valid UFI

```
<validateUFIResponse
xmlns="urn:eu:europa:ec:grow:pc:ufi:generator:types:1">
    <valid>true</valid>
</validateUFIResponse>
```

Figure 3-7: SOAP response to validateUFI for invalid UFI

```
<validateUFIResponse
xmlns="urn:eu:europa:ec:grow:pc:ufi:generator:types:1">
    <valid>false</valid>
    <error>
        <errorCode>VAL001</errorCode>
        <message>This UFI is not valid: It does not
contain 16 characters.</message>
    </error>
</validateUFIResponse>
```

3.3 Error codes

Table 3-5 and Table 3-6 list the error codes that can be returned respectively by UFI creation operations (createUFIByCount and createUFIByList) and the validation operation (validateUFI) on SOAP and REST services alike.

Table 3-5 Error code for UFI creation operations

Acronym	Definition
UFI001	Invalid or out of bound formulation number.
UFI002	Country code does not exist.
UFI003	Not a valid VAT for country.
UFI004	VAT number should be accompanied with country code.
UFI005	VAT number cannot be empty when combined with country code.
UFI006	No country found that corresponds to country group and code.
UFI007	No file has been attached to the request.
UFI008	This extension is not supported.
UFI009	Unexpected error while parsing file.
UFI010	Formulation number count is invalid.
UFI011	Request contains an invalid parameter.

Table 3-6 Error codes for UFI validation operations

Acronym	Definition
VAL001	This UFI is not valid: It does not contain 16 characters.
VAL002	This UFI is not valid: It contains at least one invalid character.
VAL003	This UFI is not valid: You may have inverted characters or typed an incorrect character.
VAL004	This UFI is not valid: Its internal country code is not correct.
VAL005	This UFI is not valid: Its internal version number is not correct.

Annex A. Examples of UFI algorithm usage

3.4 UFI with Irish VATIN

This example shows how a UFI is created for

- VATIN IE9Z54321Y (i.e. VAT country code IE and VAT number 9Z54321Y)
- Formulation number 134217728

Step 1

Determine binary representations of the UFI payload numerical value using,

- The country group code for IE from Table 2-1

  → G = 5 = 0101b (on 4 bits)

- The number of bit for the country code from Table 2-1

  → B = 7

- The country code for IE from Table 2-1

  → C = 2 = 0000010b (using B = 7 bits)

- The numerical value associated with the VAT number 9Z54321Y using the rule for IE in Table 2-2 (such conversion is necessary because Irish VAT number contain alphabetic characters; it is not required for all VAT numbers, see Table 2-2 for cases where it applies)

  → N = (26 . 25 + 24) . 10^6 + 954321 = 674954321 = 00001010000011101011111100010100001b (using 41 – 7 = 34 bits)

- The formulation number

  → F = 134217728 = 10000000000000000000000000000b (on 28 bits)

Build the complete binary representation of the UFI payload numerical value concatenating the data above (and version bit 0)

10000000000000000000000000000 0101 0000010 00001010000011101011111100010100001 0

That is, **94447329877995923682 90** in decimal.

Step 2

Convert the UFI payload numerical value to base-31 (the decimal value of each digit in base-31 is shown here),

→ 12, 14, 24, 25, 25, 19, 6, 2, 22, 20, 6, 12, 25, 27, 1

Giving, in the character set in Table 2-3, **DFSTTM62QN6DTV1**

Step 3

Reorganise the characters following Table 2-4, thus giving **MTT2SQN6FDD6TV1**

Step 4

Calculate the checksum from the reorganised representation.

Note, to ease the comprehension of this step, that the reorganised decimal representation of the digits from step 2 and 3 is

→ 19, 25, 25, 2, 24, 22, 20, 6, 14, 12, 12, 6, 25, 27, 1

Let's calculate the weighted sum of the digits (weights in red)

→ 2 . 19 + 3 . 25 + 4 . 25 + 5 . 2 + 6 . 24 + 7 . 22 + 8 . 20 + 9 . 6 +

10 . 14 + 11 . 12 + 12 . 12 + 13 . 6 + 14 . 25 + 15 . 27 + 16 . 1

= 2000

The checksum value is

→ (31 – 2000 modulo 31) modulo 31

= 15

That is, the checksum is G in the character set in Table 2-3.

After prepending the checksum the complete UFI is thus **GMTT-2SQN-6FDD-6TV1**

Checksum validation

The checksum can be verified by calculating the weighted sum of the (reorganised) digits, including the checksum digit with weight 1. The modulo 31 of this sum must be equal to 0.

That is, reusing the value of the weighted sum of step 4 and the checksum value 15,

→ (15 + 2000) modulo 31

= 0

Hence the UFI is valid with respect to its checksum.

3.5 UFI with company key

This example shows how a UFI is created for

- Company key 1828639338661
  - Formulation number 156920229

Step 1

Determine binary representations of the UFI payload numerical value using,

- The country group code must be 0

  → G = 0 = 0000b (on 4 bits)

- There is no country code

  → B = 0

  → C is not applicable

- The company key is used as is

  → N = 1828639338661 = 110101001110000110110010111110000010100101b

  (on 41 bits)

- The formulation number

  → F = 156920229 = 10010101101001101001101001 01b (on 28 bits)

Build the complete binary representation of the UFI payload numerical value concatenating the data above (and version bit 0)

10010101101001101001101001010000 1101010011100001101100101111100000101001010

That is, **1104227945445873024237 8** in decimal.

Step 2

Convert the UFI payload numerical value to base-31 (the decimal value of each digit in base-31 is shown here),

→ 14, 18, 7, 1, 11, 17, 17, 6, 1, 10, 26, 3, 0, 19, 8

Giving, in the character set in Table 2-3, **FK71CJJ61AU30M8**

Step 3

Reorganise the characters following Table 2-4, thus giving **JC1671AUKF3J0M8**

Step 4

Calculate the checksum from the reorganised representation.

Note, to ease the comprehension of this step, that the reorganised decimal representation of the digits from step 2 and 3 is

→ 17, 11, 1, 6, 7, 1, 10, 26, 18, 14, 3, 17, 0, 19, 8

Let's calculate the weighted sum of the digits (weights in red)

→ 2 . 17 + 3 . 11 + 4 . 1 + 5 . 6 + 6 . 7 + 7 . 1 + 8 . 10 + 9 . 26 +
10 . 18 + 11 . 14 + 12 . 3 + 13 . 17 + 14 . 0 + 15 . 19 + 16 . 8
= 1468

The checksum value is

→ (31 – 1468 modulo 31) modulo 31
= 20

That is, the checksum is **N** in the character set in Table 2-3.

After prepending the checksum, the complete UFI is thus **NJC1-671A-UKF3-J0M8**

Checksum validation

The checksum can be verified by calculating the weighted sum of the (reorganised) digits, including the checksum digit with weight 1. The modulo 31 of this sum must be equal to 0.

That is, reusing the value of the weighted sum of step 4 and the checksum value 20,

→ (20 + 1468) modulo 31
= 0

Hence the UFI is valid with respect to its checksum.

Annex B. Sample UFIs

This annex provides UFIs generated according to the algorithm given in §2.1. These UFIs can be used to validate an implementation of the algorithm by a Poison Centre, company or third-party software vendor.

Table 3-7: Sample UFIs

VAT		Formulation number	UFI
Country code	Number		
AT	U12345678	178956970	C23S-PQ2V-AMH9-VVRF
BE	0987654321	89478485	U1JV-SUMH-N988-U751
BG	987654321	89478485 252644556	A1JV-EUD3-498W-U23R 80SW-N2UD-FGRY-F6DA
	9987654321		
CZ	81726354	156920229 15790899 268435455	HKC1-Q7N0-DKF6-7N3E 24VQ-VGV0-WF16-7WC9 W3NN-SKC4-JXSS-V4WG
	978563421		
	9785634210		
CY	12345678C	87654321	7EHW-3KC7-748V-S1RH
	12345678Y	87654321	FEHW-3KH5-X487-SNRD
DE	112358132	134217728	KMTT-DSP3-7FD7-6RWY
DK	31415926	524544	3FQU-5GP0-Y105-J64N
EE	271828182	230087533	QY3Q-327C-QDPR-EE11
FR	RF987654321	134217728	6KTT-PSK1-AFDM-KEFU
	ZY999999999	230087533	NX3Q-2263-2DP5-UQ4T
	01012345678	268435455	F3NN-3K1J-EXSK-7PHY
	10012345678	268435455	23NN-5KKE-GXSF-7MD3
GB	987654321	156920229	GJC1-S7TH-AKFK-TR8P
	999987654321	156920229	CJC1-47ES-AKFS-WTFW
	999999999999	156920229	3JC1-47ET-6KFH-WMV8
	ZY123	268435455	53NN-7KTT-1XS1-DDPH
	AB987	268435455	M3NN-7KTS-YXSK-D8UW
GR	567438921	66260700	QNWM-9X6E-E46N-G4GJ
FI	18273645	29979245	VWF9-CDT4-2S2N-PDTV

VAT		Formulation number	UFI
Country code	Number		
HR	16021765654	268435455	53NN-KKPX-SXSD-QJY7
HU	22334455	238219293	AU06-7HHD-64QN-8RHF
IE	9Z54321Y	134217728	GMTT-2SQN-6FDD-6TV1
	9+54321Y	134217728	KMTT-2SQQ-0FDQ-6A5D
	9*54321Y	134217728	GMTT-2SQR-UFD0-6TFR
	9876543Z	230087533	JY3Q-R2M8-GDP2-DQRS
	9876543ZW	230087533	XY3Q-S215-2DPF-DA4U
	9876543AB	182319099	TUG4-PE6C-4XHP-RSAM
IS	AB3D5F	182319099	XUG4-WE32-RXHD-RHWU
	1ZY2BA	268435455	53NN-1KDC-JXSH-W6WV
IT	14286244833	214783315	WK3F-PYSX-TXM0-K9AV
LI	99999	182319099	CUG4-FEHP-HXHW-SC4E
LT	987654321	15790899	W3VQ-HGW8-UF12-W7QP
	987654321098	156920229	SJC1-P7FR-DKF3-Y2YC
LU	16726218	214783315	FK3F-8YC6-1XMK-SAQ5
LV	39903127176	182319099	WUG4-5E2S-UXHN-M5C7
MT	99887766	83144621	7KW0-SMM5-2Q7N-4K8D
NL	999999999B77	96485337	QJ0V-J1JU-9Y8W-37TG
NO	958473621	268435455	63NN-7KPT-WXS8-WGYA
PL	4835978701	19621109	XRMW-9HU2-PT1U-7JNN
PT	998776554	30051977	K9WS-JKK3-WS2E-1WSC
RO	98	252644556	E0SW-U2CF-KGR6-FRKW
	9081726354	214783315	2K3F-QYHK-YXMU-RTN5
SE	987654321098	156920229	1KC1-87DH-0KFR-2WGE
SI	12345678	178956970	U23S-WQK5-AMH7-V03N
SK	9987654321	252644556	N0SW-W2AP-FGRV-F9RH
No VATIN	Company key 1828639338661	156920229	NJC1-671A-UKF3-J0M8

EUROPEAN CHEMICALS AGENCY
ANNANKATU 18, P.O. BOX 400,
FI-00121 HELSINKI, FINLAND
ECHA.EUROPA.EU