

NICC ND 1024 v1.1.1 (2008-12)

NICC Document

NGN; Number Portability Common Database; Notification and Data Download Web Service

Note : This standard is intended to allow fulfilment of changes to General Condition 18 which were announced by Ofcom via the November 2007 Statement entitled "Telephone number portability for consumers switching suppliers - Concluding Statement" and subsequently set aside by the Competition Appeal Tribunal (Case 1094/3/3/08). At the time of publishing, Ofcom is reviewing the regulation in this area so readers are cautioned that changes to this standard may prove necessary.

Network Interoperability Consultative Committee,
Ofcom,
2a Southwark Bridge Road,
London,
SE1 9HA.

© 2008 Ofcom copyright

NOTICE OF COPYRIGHT AND LIABILITY

Copyright

All right, title and interest in this document are owned by Ofcom and/or the contributors to the document unless otherwise indicated (where copyright be owned or shared with a third party). Such title and interest is protected by United Kingdom copyright laws and international treaty provisions.

The contents of the document are believed to be accurate at the time of publishing, but no representation or warranty is given as to their accuracy, completeness or correctness. You may freely download, copy, store or distribute this document provided it is not modified in any way and it includes this copyright and liability statement.

You may not modify the contents of this document. You may produce a derived copyright work based on this document provided that you clearly indicate that it was created by yourself and that it was derived from this document and provided further that you ensure that any risk of confusion with this document is avoided.

Liability

Whilst every care has been taken in the preparation and publication of this document, NICC, nor any committee acting on behalf of NICC, nor any member of any of those committees, nor the companies they represent, nor any person contributing to the contents of this document (together the “Generators”) accepts liability for any loss, which may arise from reliance on the information contained in this document or any errors or omissions, typographical or otherwise in the contents.

Nothing in this document constitutes advice. Nor does the transmission, downloading or sending of this document create any contractual relationship. In particular no licence is granted under any intellectual property right (including trade and service mark rights) save for the above licence to copy, store and distribute this document and to produce derived copyright works.

The liability and responsibility for implementations based on this document rests with the implementer, and not with any of the Generators. If you implement any of the contents of this document, you agree to indemnify and hold harmless the Generators in any jurisdiction against any claims and legal proceedings alleging that the use of the contents by you or on your behalf infringes any legal right of any of the Generators or any third party.

None of the Generators accepts any liability whatsoever for any direct, indirect or consequential loss or damage arising in any way from any use of or reliance on the contents of this document for any purpose.

If you have any comments concerning the accuracy of the contents of this document, please write to:

The Technical Secretary,
Network Interoperability Consultative Committee,
Ofcom,
2a Southwark Bridge Road,
London SE1 9HA.

Contents

Intellectual Property Rights	5
Foreword	5
Introduction	5
1 Scope	6
2 References	6
2.1 Normative references	6
3 Definitions and abbreviations	7
3.1 Definitions	7
3.2 Abbreviations	7
3.3 Usage indications	7
4 Notification and Data Download Service description	7
4.1 Notifications (D ₁ Reference Point)	7
4.2 Downloads (D ₂ Reference Point)	8
4.2.1 Full Download	8
4.2.2 Incremental Download	8
5 Namespaces	9
5.1 Notify Interface	9
5.2 Data Download Interface	9
6 Web Service interface definition	9
6.1 Notification Service	9
6.1.1 notificationRequest element	10
6.1.2 notificationResponse element	10
6.1.3 Referenced faults	10
6.2 Data Download Service	10
6.2.1 downloadRequest element	10
6.2.2 downloadResponse element	10
6.2.3 Referenced faults	11
7 Definition of types and elements	11
7.1 checkpointType ‡	11
7.2 download element	11
7.3 downloadModeType	11
7.4 ecc element	11
7.5 entry element	12
7.6 entryIDType	12
7.7 fullEcc element	12
7.8 idType ‡	13
7.9 ims element ‡	13
7.10 notify element	13
7.11 pstn element ‡	13
7.12 sectionData element	13
7.13 sectionType ‡	14
7.14 send-n element	14
8 Fault definitions	14
8.1 Fault number ranges by service	14
8.2 ServiceException	15
8.3 PolicyException	15
9 Generation of ECCs	15
9.1 General issues	15
9.2 Details	15
9.2.1 Canonicalisation	15

9.2.2	Batching	16
9.2.3	Hashing	16
9.2.4	Consolidation	17
Annex A (normative): WSDL for common data definitions.....		18
A.1	WSDL for the Notification Service	18
A.2	WSDL for the Data Download Service.....	20
A.3	Definitions also used in ND1025	22
A.3.1	Common definitions	22
A.3.2	Definitions specific to the Data Download Service	23
A.4	WSDL relating to faults.....	23
A.4.1	Fault elements	23
A.4.2	Fault messages	24
A.4.3	Fault operations.....	24
Annex B (informative): Example messages.....		25
B.1	NotifyRequest message.....	25
B.2	NotifyResponse message	25
B.3	DownloadRequest message	26
B.4	DownloadResponse message.....	26
Annex C (informative): ECC rationale		27
History		28

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to NICC.

Pursuant to the NICC IPR Policy, no investigation, including IPR searches, has been carried out by NICC. No guarantee can be given as to the existence of other IPRs which are, or may be, or may become, essential to the present document.

Foreword

This NICC Document (ND) has been produced by NICC TSG/NNA

Introduction

In November 2007 Ofcom published a Statement entitled “Telephone number portability for consumers switching suppliers” – “Concluding Statement”. In that statement Ofcom issued a Notification of modifications to General Condition 18. In that modification Communications Providers are required to establish a “Common Database” for the purposes of Number Portability. The present document sets out to fulfil the need to describe a technical solution for that “Common Database” irrespective of how and by whom it is used.

The present document has been produced by the NICC Naming Numbering and Addressing Working Group.

Nothing in this document shall be taken as preventing the further elaboration of the specification contained herein for the purposes of specifying systems that are intended to be embodied in an implementation. Such further elaboration may be documented elsewhere and make reference to the present document.

1 Scope

The present document is a Stage 3 protocol description for the distribution of data within the NICC Common Numbering Database described in ND1631 [2]. This document describes the specific information flows across Reference Points D₁ and D₂ of ND1631 [2] over the generic Web Services described in ND1023 [3].

2 References

For the particular version of a document applicable to this release see [ND1610](#) [10].

NOTE: While any hyperlinks included in this clause were valid at the time of publication NICC cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are indispensable for the application of this document.

NOTE: While any hyperlinks included in this clause were valid at the time of publication NICC cannot guarantee their long term validity.

[1] W3C Recommendation (2 May 2001): "XML Schema Part 2: Datatypes".

NOTE: Available at: <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.

[2] NICC ND1631 NGN; PSTN/ISDN Service Interconnect; Architecture for usage of Common Numbering Database

[3] NICC ND1023 NGN; Number Portability Common Database; Web Services Communication & Common XML Features

[4] NICC ND1025 NGN; Number Portability Common Database; Management Web Service

[5] IETF STD 13: "Domain Name System".

NOTE: Available at: <http://www.ietf.org/rfc/rfc1034.txt>.

[6] IETF RFC3174 - US Secure Hash Algorithm 1 (SHA1)

NOTE: Available at: <http://www.ietf.org/rfc/rfc3174.txt>.

[7] IETF RFC4634 – US Secure Hash Algorithms (SHA and HMAC-SHA)

NOTE: Available at: <http://www.ietf.org/rfc/rfc4634.txt>.

[8] NICC ND1633: "UK Next Generation Networks; Element Naming Framework".

[9] W3C Recommendation (2 May 2001): "XML Schema Part 2: Datatypes".

NOTE: Available at: <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.

[10] NICC, ND1610, Multi-Service Interconnect of UK Next Generation Networks

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Web Service: software system designed to support interoperable machine-to-machine interaction over a network

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in ND1631 [2] and the following apply:

CDB	Central DataBase
IETF	Internet Engineering Task Force
W3C	World Wide Web Consortium
WSDL	Web Service Definition Language
XML	Extensible Markup Language

3.3 Usage indications

In the tables in this document defining data structures, the following codes are used in the “Usage” column:

D	Occurs or not depending on what data is to be transmitted; see the description
M	Mandatory
O	Optional at choice of entity creating the message
R	May be repeated any number of times according to the data being transmitted
MR	occurs one or more times
OR	occurs zero or more times

4 Notification and Data Download Service description

The requirements for this service are defined in NICC ND1631 [2]. The services described here implement the information flows across Reference Points D₁ and D₂. This clause is provided for guidance and, in the event of any discrepancy between it and ND1631, the latter is definitive.

The overall function provided by these information flows is to provide an update mechanism for the Common Database so that the contents held in the Central Database (CDB) may be propagated in a timely manner to all its replicas.

The data held in the CDB about numbers is split into a number of Sections each of which is characterised by a telephone number prefix. The overall architecture provides that the CDB is told which servers are to receive notifications of changes to Sections within the database.

4.1 Notifications (D₁ Reference Point)

When changes occur in a particular Section of the database the CDB notifies the CP’s servers of the change and gives the change a checkpoint value. For any Section the checkpoint value is an integer that is different for each notification with later notifications having higher values than earlier ones. The values increase with time such that equal differences in checkpoint values will correspond to equal differences in time between notifications.

On receiving a notification the CP’s server responds so that the CDB is able to be assured that the server is receiving notifications. It is possible in the case of message loss that there will be multiple notifications with the same data. For each the CP server shall respond normally. If a fault persists the CDB will stop sending notifications to the server that is determined as faulty. Because notifications may be sent over a separate connection to download requests, it is possible that a Notify may arrive with an earlier Checkpoint value than the most recent Download.

Notification is sent from the CDB as a Notification Request with a Notification Response from the CP server and is a single operation.

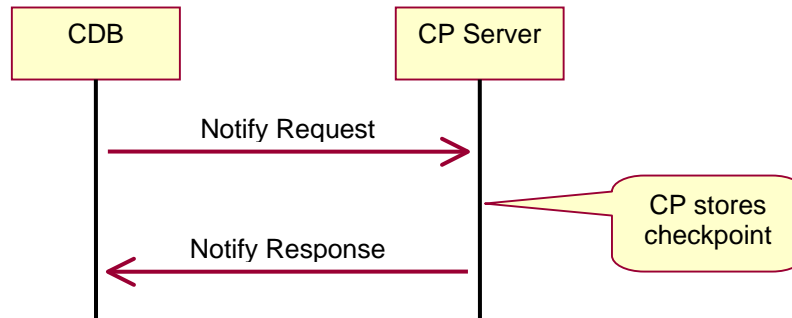


Figure 1 – message flow for Notifications

4.2 Downloads (D₂ Reference Point)

Download Requests are sent from the CP Server as either a Full Download Request or an Incremental Download Request and may (but need not) be initiated following the Notification service. The CDB will respond with a Download Response. The response to a Full Download Request must contain full data, but that to an Incremental Download Request may contain either full or incremental data; the data structure is the same for each. The CP Server should process the data in the order it occurs in the download; if a number occurs more than once, it should be processed each time it occurs. The data in the response includes a Checkpoint value that is stored as the last valid Checkpoint on successful receipt of the data. It is possible that this value is more recent than the Checkpoint in the most recent Notify.

4.2.1 Full Download

When a CP needs to receive all the data within a given Section it uses the Full Download request. A Full Download Response is always returned in response.

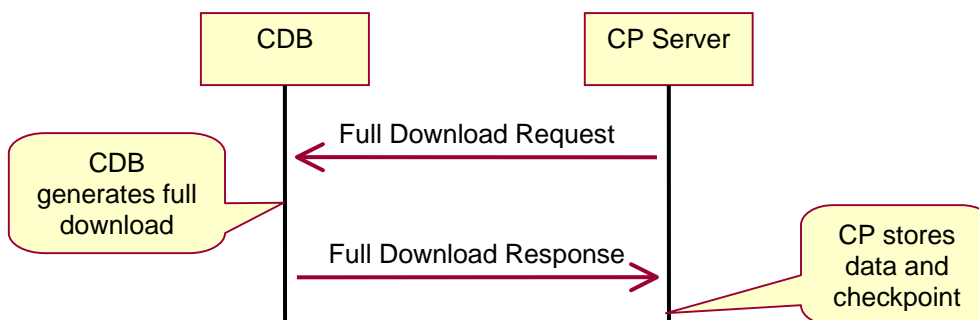


Figure 2 – message flow for Full Download

4.2.2 Incremental Download

When a CP determines that its data is not up-to-date for a given Section (e.g. following the receipt of one or more notifications for the Section) it may send an Incremental Download request along with the checkpoint value

corresponding to the last set of data that was successfully downloaded. The data provided for an incremental download contains each successive change for all affected numbers in the Section and, where multiple changes have occurred, the resulting data may exceed the quantity of data in a full download. Therefore the response from the CDB will vary based upon the quantity of data involved and the CDB will send the data as either a Full Download Response or an Incremental download Response, whichever form will require less data.

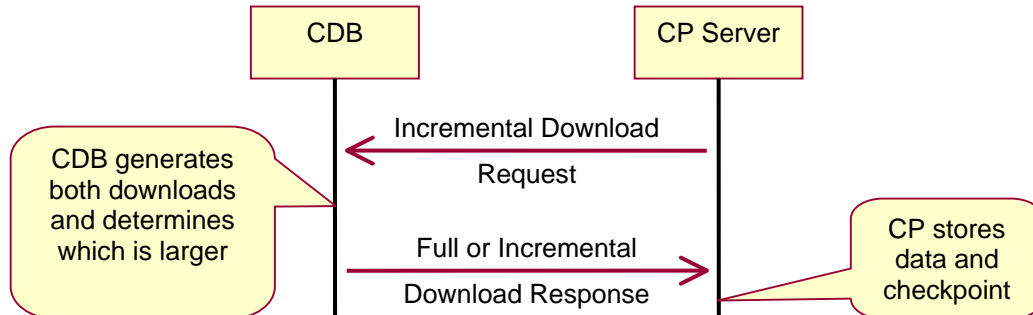


Figure 3 – message flow for Incremental Download

5 Namespaces

The reason for having namespaces is described in ND1023 [3]. When an identifier in this document does not have a namespace prefix, it means the identifier defined in the appropriate one of the following namespaces. The namespace “xs” refers to the namespace defined in [9]. Other namespaces are defined in ND1023 [3].

5.1 Notify Interface

The Notify interface uses the namespace:

http://www.uktel.org.uk/wsd/nd1024/notify/v1_1

5.2 Data Download Interface

The download interface uses the namespace:

http://www.uktel.org.uk/wsd/nd1024/download/v1_1

6 Web Service interface definition

This clause describes the specific messages that provide the Notification and Data Download Services. They make use of data types described in clause 7. A WSDL representation of this interface specification is provided in Annex A. Where there is a conflict, the WSDL representation is definitive. Examples of these messages are provided in Annex B.

6.1 Notification Service

This service provides the ability for the CDB to notify the CP that a set of changes has been made to one or more Sections of the common database. The CDB sends a **NotifyRequest** message to the CP, the body of which is a

notificationRequest element. The CP responds with a **NotifyResponse** message, the body of which is a **notificationResponse** element, or with a fault.

6.1.1 notificationRequest element

Attribute	Attribute type	Usage	Description
ID	idType	M	Transaction Identity

Content	Content type	Usage	Description
Sections	notify	MR	Information about each Section that has been changed

6.1.2 notificationResponse element

Attribute	Attribute type	Usage	Description
ID	idType	M	Transaction Identity
Count	xs:positiveInteger	M	The number of Sections listed in the notification

The ID attribute of the response **must** be identical to that of the request. The Count attribute **must** give the number of notify elements in the request.

6.1.3 Referenced faults

These faults are described in clause 8 or ND1023 [3].

- SVC0001 – Service error.
- SVC0002 – Invalid input value.
- POL0001 – Policy error.
- POL0002 – Subscription Error

6.2 Data Download Service

This service provides the ability for the CP to request a download from the CDB. The CP sends a **DownloadRequest** message to the CDB, the body of which is a **downloadRequest** element. The CDB responds with a **DownloadResponse** message, the body of which is a **downloadResponse** element, or with a fault.

The Information flows in ND1631 [2] show different flows for incremental and full downloads. However, at the protocol level the requests and responses are identical, using a type indicator within the download element to show which is intended.

6.2.1 downloadRequest element

Attribute	Attribute type	Usage	Description
ID	idType	M	Transaction Identity

Content	Content type	Usage	Description
Sections	download	MR	A list of Sections to be downloaded

6.2.2 downloadResponse element

Attribute	Attribute type	Usage	Description
ID	idType	M	Transaction Identity

Content	Content type	Usage	Description
Sections	sectionData	MR	The data for a Section of the database

The ID attribute **must** be identical to that of the request. There **must** be one sectionData element for each download element in the request, though not necessarily in the same order, and at least one **must not** be of type “omitted”.

6.2.3 Referenced faults

These faults are described in clause 8 or ND1023 [3].

- SVC0001 – Service error.
- SVC0002 – Invalid input value.
- POL0001 – Policy error.
- POL0002 – Subscription Error

7 Definition of types and elements

The messages defined in clause 6 make use of the following types and elements. These are listed in alphabetical order.

The type names of elements defined in this clause consist of the element name with “Type” suffixed.

Those types and elements marked with a double dagger (‡) are also used in ND1025 [4].

7.1 checkpointType ‡

The checkpointType type represents a checkpoint reference. It is an unsigned 32-bit integer (i.e. an integer between 0 and 4 294 967 295 inclusive).

7.2 download element

The download element indicates which Section of the CDB is being requested, and whether an incremental or full download is desired.

Attribute	Attribute type	Usage	Description
Section	sectionType	M	The Section of the CDB that has changed
Type	downloadModeType	M	Type of download requested (full or incremental)
Checkpoint	checkpointType	D	The checkpoint reference associated with the request; required for incremental requests, forbidden for full requests

7.3 downloadModeType

The downloadModeType type indicates whether a download request or response is an incremental one or a full one, or whether the Section has been omitted from the download (for example a Section could be omitted if the overall response would otherwise be too large to be practical). It is one of the strings “incremental”, “full”, or “omitted”.

7.4 ecc element

The ecc element represents error checking codes for a data download; that is, sufficient data to determine whether the download has been corrupted.

Attribute	Attribute type	Usage	Description
Algorithm	xs:token	M	The ecc hash algorithm used.
Value	xs:hexBinary	M	The ecc of the downloaded data.

The ECC is calculated using the procedure given in clause 9 which, in turn, makes use of the hash algorithm specified in the Algorithm attribute. This procedure generates a sequence of hexadecimal digits which form the value of the Value attribute. The ECC is calculated over the data in the download.

7.5 entry element

The entry element describes the CDB content for one or more telephone numbers or telephone number prefixes.

Attribute	Attribute type	Usage	Description
first	entryIDType	M	The first number in the range.
last	entryIDType	O	The last number in the range; if omitted, the range contains only one number.
ttl	xs:unsignedInt	M	The Time To Live, in seconds, of the data.

Content	Content type	Usage	Description
Send-N	sendNType	D	Send-N structure. See NOTE 1
PSTN	pstnType	D	PSTN destination group. See NOTE 1
IMS	imsType	D	Sip url without the userinfo part and omitting the default prefix for this Section. See NOTE 1.

NOTE 1: There will be either none of these three elements, exactly one Send-N element, or either or both of the PSTN or IMS elements. If both are present, the PSTN element must occur before the IMS element.

If the last attribute is omitted, this element applies to a single telephone number or prefix. If the attribute is provided, then the element applies to a consecutive range of numbers or prefixes; the last attribute **must** be the same length as and numerically equal to or greater than the first attribute.

The ttl attribute **must** be no greater than 2 147 483 647 (i.e. it is an unsigned 31-bit integer).

The contents of the element replace any existing data for that number, prefix, or range, and are one of:

- nothing
- a send-n element
- either or both of the pstn and ims elements, in that order

7.6 entryIDType

The entryIDType type represents the part of a telephone number or prefix after the Section number (that is, the D digit onwards). It consists of a sequence of between 1 and 6 decimal digits.

7.7 fullEcc element

The fullEcc element represents error checking codes for a version of an entire Section stored in the CDB.

Attribute	Attribute type	Usage	Description
Algorithm	xs:token	M	The ecc hash algorithm used.
Checkpoint	checkpointType	M	The checkpoint reference of the data the ecc applies to
Value	xs:hexBinary	M	The ecc of the Section data.

The Checkpoint attribute indicates which version of the Section data the ECC applies to. It **must** be either the checkpoint in the enclosing sectionData element or the checkpoint in the download element which requested this download.

The ECC is calculated using the procedure given in clause 9 which, in turn, makes use of the hash algorithm specified in the Algorithm attribute. This procedure generates a sequence of hexadecimal digits which form the value of the Value attribute. The ECC is calculated over all the data in the Section.

7.8 idType ‡

The idType type represents a transaction identity. This **must** be unique for each new transaction. The same identity as used in a request **must** be used in its response and in any retransmission of either request or response.

A transaction identity is a string of between 1 and 32 lowercase letters, digits, and hyphens. It **must** begin with a <provider> which identifies the CP sending or receiving the initial request, as defined in ND1633 [8], followed by a hyphen; the sender of the request shall determine the form of the remainder of the transaction identity.

7.9 ims element ‡

The ims element holds the IMS Destination Group.

Content	Content type	Usage	Description
IMS DG	xs:token	M	The IMS destination group.

The IMS Destination Group is a domain name. If the value ends in a dot, it is used unaltered. Otherwise the value of the IMSsuffix attribute of the surrounding downloadResponse element is appended to it to create the full domain name.

Note: it is not significant whether the full name was provided directly or generated by appending the suffix.

7.10 notify element

The notify element holds information about a Section of the database that has changed.

Attribute	Attribute type	Usage	Description
Section	sectionType	M	The Section of the CDB that has changed
Checkpoint	checkpointType	M	The new checkpoint reference available for download

7.11 pstn element ‡

The pstn element holds the PSTN Destination Group.

Content	Content type	Usage	Description
PSTN DG	xs:token	M	The PSTN destination group.

The PSTN Destination Group is an 8-digit number in the range 72000000 to 79999999.

7.12 sectionData element

The sectionData element holds the data and ECC for a single Section of the database, or indicates that it is omitted.

Attribute	Attribute type	Usage	Description
Section	sectionType	M	The Section of the CDB that has changed
Type	downloadModeType	M	Type of download (full, incremental, or omitted)
Checkpoint	checkpointType	M	The new checkpoint reference associated with the download. In the case of an omitted Section, this is the latest checkpoint held by the CDB.
IMSsuffix	xs:token	O	Suffix to be appended when IMS data for a number uses a relative domain name (that is, one not ending in a dot); default value is “.uktel.org.uk.”.

Content	Content type	Usage	Description
Entries	Entry	OR	The data of that Section of the database
Data ECCs	Ecc	OR	Error correcting code information for the downloaded data
Full ECCs	fullEcc	OR	Error correcting code information for data stored in the CDB

The IMSsuffix **must** be a domain name preceded by and ending with a dot. The order of the Entries is not significant. No two Entries shall refer to the same telephone number or prefix. The ECCs **must** occur after the entries, and any full ECCs must occur after any data ECCs, but the order of the ECCs within each type is not significant. If the type is “omitted” there **must not** be any contents; otherwise there **must** be at least one data ECC. A full ECC **must not** be provided in any download other than incremental.

Note: the IMSsuffix only affects the meaning of any enclosed ims element. Once it has been applied to those elements, it need not be stored; a subsequent download with a different value does *not* affect the meaning of previous downloads. The sole purpose of the value is to reduce the size of the download by factoring out the most common suffix.

7.13 sectionType ‡

The sectionType type represents a Section of the CDB. It is a five digit sequence in the range 01000 to 09999.

7.14 send-n element

The send-n element holds SEND-N information. This is represented as the number of digits in the shortest valid E.164 number that begins with the existing number prefix.

Content	Content type	Usage	Description
Send-N data	send-nType	M	The send-N data.

The SEND-N information is a decimal number between 4 and 15, with no leading zero.

NOTE: all valid UK numbers have a length between 9 and 12 inclusive. The wider range has been left in to allow for other uses in the future.

NOTE: for example, the UK number 0141 496 0223 has 12 digits in E.164 form (44 141 496 0223) and so is represented by the value 12.

8 Fault definitions

8.1 Fault number ranges by service

ServiceException and PolicyException message numbers 0200 to 0299 are reserved in ND1023 [3] for the Notification and Data Download Web Service. The following table includes fault number ranges that are reserved for use by specific CDB Download Web Services.

Web Service	SVC and POL ranges
Notification	0200 to 0219
Data Download	0220 to 0239
Reserved for future	0240 to 0299

8.2 ServiceException

No service-specific messages are defined at present.

8.3 PolicyException

No service-specific messages are defined at present.

9 Generation of ECCs

This clause describes the algorithm used to generate ECCs of the data being downloaded. A discussion of the issues in its design may be found in Annex C.

The algorithm consists of four stages:

1. The data to be checked is canonicalised. This is described in clause 9.2.1.
2. The data is “batched”. This is described in clause 9.2.2.
3. Each batch is hashed. This is described in clause 9.2.3.
4. The hash values are consolidated. This is described in clause 9.2.4.

The bit-string resulting from the last stage is converted to a string of hexadecimal digits which forms the ECC (if the length of the bit-string is not a multiple of 4 bits, up to 3 zero bits are added to the left end to make it so).

9.1 General issues

For an ecc element, the algorithm of this Section is applied to the data in the entry elements within the sectionData element that contains the ECC. For a fullEcc element, it is applied to the data that would occur in the entry elements of a full download of the indicated version of the appropriate Section of the database.

The ecc or fullEcc element contains an Algorithm attribute. This **shall** contain a name representing a batching level and hashing algorithm. Each user of the CDB shall agree which combinations shall be used in downloads requested by that user (each combination creates a separate element). The CDB shall offer at least the following:

Name	Batching level	Hashing algorithm
0 : SHA1	0	SHA1 [6]
0 : SHA256	0	SHA256 [7]

This table will be reviewed as necessary.

9.2 Details

9.2.1 Canonicalisation

The data is processed as follows to generate a set of text lines. The lines are then sorted lexicographically (that is, 9602 comes between 960199 and 960200).

- Each telephone number or prefix shall generate a single line. If an entry element contains a “last” attribute different to its “first” attribute, it generates multiple lines.
- The line consists of up to six fields separated by single spaces:

1. the Section number;
2. the telephone number or prefix;
3. the time-to-live value;
4. the PSTN destination group;
5. the IMS destination group, with any default prefix restored;
6. the SEND-N information.

Where the last field or fields are empty, the spaces are omitted as well (that is, a line never ends with a space)

- Each line ends with a CR octet (value 13) followed by an LF octet (value 10).

For example, the following downloadResponse element:

```
<downloadResponse ID="123457" Section="01414" Type="incremental"
Checkpoint="805131250">
  <entry first="960224" last="960226" ttl="720">
    <pstn>74016285</pstn>
    <ims>cs3.p28.dg.example</ims>
  </entry>
  <entry first="960227" ttl="720" />
  <entry first="9600" last="9603" ttl="720">
    <send-n>12</send-n>
  </entry>
  <entry first="960223" ttl="720">
    <pstn>73100244</pstn>
  </entry>
</downloadResponse>
```

will create the following lines (\$ is used to represent the trailing CR LF and _ to represent some spaces):

```
01414 9600 720 _ 12$
01414 9601 720 _ 12$
01414 9602 720 _ 12$
01414 960223 720 73100244$
01414 960224 720 74016285 cs23.p28.dg.example.uktel.org.uk$
01414 960225 720 74016285 cs23.p28.dg.example.uktel.org.uk$
01414 960226 720 74016285 cs23.p28.dg.example.uktel.org.uk$
01414 960227 720$
01414 9603 720 _ 12$
```

9.2.2 Batching

The text lines generated by the previous stage are divided into batches based on the “batching level”. This shall be a number between 0 and 4 inclusive. If the batching level is b , then each batch consists of all lines which have the same first b digits in the telephone number or prefix field (if b is 0, then all the lines form a single batch). Thus the example data in the previous sub-clause consists of one batch if the batching level is between 0 and 3 inclusive, and 4 batches (with 1, 1, 6, and 1 lines respectively) if the batching level is 4.

If there are any lines where the telephone number or prefix is shorter than the batching level (e.g. a prefix “96” with a batching level of 3 or 4) then these lines together form a single additional batch.

9.2.3 Hashing

Each batch is hashed using the selected hashing algorithm. This algorithm **shall** generate a bit-string (that is, a string of binary digits). If the algorithm is one with a selectable initial vector or other state, this shall be specified as part of its description.

Empty batches shall not be generated or, equivalently, shall be given a hash of “0” irrespective of what the hashing algorithm would generate with no data.

9.2.4 Consolidation

The hash values generated by each batch are consolidated into a single value. If the values are not all the same length, the shorter ones shall be extended to match the longest by adding zero digits at the left hand end. Then the values shall be XORed together to generate the final value.

Note: XOR is done on a binary digit-by-digit basis and (since it is both commutative and associative) the order that the values are XORed is irrelevant.

Note: The overall effect of this algorithm is that the hash of each batch can be stored and, when the CDB contents change, only those batches that have changed need to be re-hashed to generate the new ECC. This eliminates the requirement to hash the entire contents of a Section every time a single entry changes. The use of XOR for the final combination means that the new ECC is the XOR of the old ECC, the old hashes of the changed Sections, and the new hashes of those Sections.

Annex A (normative): WSDL for common data definitions

This document/literal WSDL representation of this interface specification is compliant to the content requirements specified in the present document.

A.1 WSDL for the Notification Service

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Last modified: 2008-09-15T16:49:30Z -->
<definitions name="NotifyDefinition"
  xmlns:nd1024="http://www.uktel.org.uk/wsd1/nd1024/notify/v1_1"
  xmlns:soap="http://schemas.xmlsoap.org/wsd1/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns="http://schemas.xmlsoap.org/wsd1/"
  targetNamespace="http://www.uktel.org.uk/wsd1/nd1024/notify/v1_1">

  <types>
    <schema
      xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.uktel.org.uk/wsd1/nd1024/notify/v1_1"
    >
```

This WSDL should be read as if the contents of A.3.1 were inserted here.

This WSDL should be read as if the contents of A.4.1 were inserted here.

```
    <element name="notificationRequest">
      <complexType>
        <attribute name="ID" type="nd1024:idType" use="required" />
        <element name="notify" type="nd1024:notifyType"
          maxOccurs="unbounded" />
      </complexType>
    </element>

    <element name="notificationResponse">
      <complexType>
        <attribute name="ID" type="nd1024:idType" use="required" />
        <attribute name="Count" type="positiveInteger" use="required" />
      </complexType>
    </element>

    <complexType name="notifyType">
      <attribute name="Section" type="nd1024:sectionType" use="required" />
      <attribute name="Checkpoint" type="nd1024:checkpointType"
        use="required" />
    </complexType>

  </schema>
</types>
```

```

<message name="NotifyRequestMessage">
  <part name="NotifyRequest" element="nd1024:notificationRequest" />
</message>
<message name="NotifyResponseMessage">
  <part name="NotifyResponse" element="nd1024:notificationResponse" />
</message>

```

This WSDL should be read as if the contents of A.4.2 were inserted here.

```

<portType name="NotifyPortType">
  <operation name="NotifyOperation">
    <input message="notify:NotifyRequestMessage" />
    <output message="nd1024:NotifyResponseMessage" />
  </operation>
</portType>

```

This WSDL should be read as if the contents of A.4.3 were inserted here.

```

</operation>
</portType>

<binding name="NotifyBinding" type="nd1024:NotifyPortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="NotifyOperation">
    <soap:operation soapAction="notify" />
    <input>
      <soap:body use="literal" namespace="http://www.uktel.org.uk"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="literal" namespace="http://www.uktel.org.uk"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
    <fault name="ServiceException">
      <soap:fault name="ServiceException" use="literal" />
    </fault>
    <fault name="PolicyException">
      <soap:fault name="PolicyException" use="literal" />
    </fault>
  </operation>
</binding>

<service name="Notify">
  <port name="NotifyPort" binding="nd1024:NotifyBinding">
    <soap:address location="http://localhost/D-reference-point/Notify" />
  </port>
</service>
</definitions>

```

A.2 WSDL for the Data Download Service

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Last modified: 2008-09-30T17:46:59Z -->
<definitions name="CDBDefinitions"
  xmlns:download="http://www.uktel.org.uk/wsdl/nd1024/download/v1_1"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://www.uktel.org.uk/wsdl/nd1024/download/v1_1">

  <types>
    <schema
      xmlns="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://www.uktel.org.uk/wsdl/nd1024/download/v1_1">
```

This WSDL should be read as if the contents of A.3.1 were inserted here.

This WSDL should be read as if the contents of A.3.2 were inserted here.

This WSDL should be read as if the contents of A.4.1 were inserted here.

```
  <element name="downloadRequest">
    <complexType>
      <attribute name="ID" type="download:idType" use="required" />
      <element name="download" type="download:downloadType"
        maxOccurs="unbounded" />
    </complexType>
  </element>

  <element name="downloadResponse">
    <complexType>
      <attribute name="ID" type="nd1024:idType" use="required" />
      <element name="sectionData" type="nd1024:sectionDataType"
        maxOccurs="unbounded" />
    </complexType>
  </element>

  <simpleType name="downloadModeType">
    <restriction base="token">
      <enumeration value="full" />
      <enumeration value="incremental" />
      <enumeration value="omitted" />
    </restriction>
  </simpleType>

  <complexType name="downloadType">
    <attribute name="Section" type="nd1024:sectionType" use="required" />
    <attribute name="Type" type="nd1024:downloadModeType" use="required" />
    <attribute name="Checkpoint" type="nd1024:checkpointType"
      use="optional" />
  </complexType>

  <complexType name="eccType">
    <attribute name="Algorithm" type="token" use="required" />
    <attribute name="Value" type="hexBinary" use="required" />
  </complexType>
```

```

<simpleType name="entryIDType">
  <restriction base="token">
    <pattern value="[0-9]{1,6}" />
  </restriction>
</simpleType>

<complexType name="entryType">
  <choice>
    <element name="send-n" type="nd1024:send-nType" />
    <sequence>
      <element name="pstn" type="nd1024:pstnType" minOccurs="0" />
      <element name="ims" type="nd1024:imsType" minOccurs="0" />
    </sequence>
  </choice>
  <attribute name="first" type="nd1024:entryIDType" use="required" />
  <attribute name="last" type="nd1024:entryIDType" use="optional" />
  <attribute name="ttl" type="unsignedInt" use="optional" default="0" />
</complexType>

<complexType name="fullEccType">
  <attribute name="Algorithm" type="token" use="required" />
  <attribute name="Value" type="hexBinary" use="required" />
  <attribute name="Checkpoint" type="nd1024:checkpointType"
    use="required" />
</complexType>

<complexType name="sectionDataType">
  <attribute name="Section" type="nd1024:sectionType" use="required" />
  <attribute name="Type" type="nd1024:downloadModeType" use="required"/>
  <attribute name="Checkpoint" type="nd1024:checkpointType"
    use="required" />
  <attribute name="IMSSuffix" type="token" use="optional"
    default=".uktel.org.uk" />
  <sequence>
    <element name="entry" type="nd1024:entryType"
      minOccurs="0" maxOccurs="unbounded" />
    <element name="ecc" type="nd1024:eccType"
      minOccurs="1" maxOccurs="unbounded" />
    <element name="fullEcc" type="nd1024:fullEccType"
      minOccurs="0" maxOccurs="unbounded" />
  </sequence>
</complexType>

<simpleType name="sendnType">
  <restriction base="positiveInteger">
    <minInclusive="4" />
    <maxInclusive="15" />
    <pattern value="[1-9][0-9]?" />
  </restriction>
</simpleType>

</schema>
</types>

<message name="DownloadRequestMessage">
  <part name="DownloadRequest" element="nd1024:downloadRequest" />
</message>
<message name="DownloadResponseMessage">
  <part name="DownloadResponse" element="nd1024:downloadResponse" />
</message>

```

This WSDL should be read as if the contents of A.4.2 were inserted here.

```

<portType name="DownloadPortType">
  <operation name="DownloadOperation">
    <input message="nd1024:DownloadRequestMessage" />
    <output message="nd1024:DownloadResponseMessage" />

```

This WSDL should be read as if the contents of A.4.3 were inserted here.

```

  </operation>
</portType>

<binding name="DownloadBinding" type="nd1024:DownloadPortType">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="DownloadOperation">
    <soap:operation soapAction="Download" />
    <input>
      <soap:body use="literal" namespace="http://www.uktel.org.uk"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </input>
    <output>
      <soap:body use="literal" namespace="http://www.uktel.org.uk"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
    <fault name="ServiceException">
      <soap:fault name="ServiceException" use="literal" />
    </fault>
    <fault name="PolicyException">
      <soap:fault name="PolicyException" use="literal" />
    </fault>
  </operation>
</binding>

<service name="DataDownload">
  <port name="DataDownloadPort" binding="nd1024:DataDownloadBinding">
    <soap:address location="http://localhost/D-reference-point/Download" />
  </port>
</service>
</definitions>

```

A.3 Definitions also used in ND1025

These definitions are also used in ND1025 [4].

A.3.1 Common definitions

These definitions are used by both the Notification Service and the Data Download Service.

```

<simpleType name="checkpointType">
  <restriction base="unsignedInt" />
</simpleType>

<simpleType name="idType">
  <restriction base="token">
    <maxLength value="32" />
    <pattern value="[a-z0-9]+-[a-z0-9-]+" />
  </restriction>
</simpleType>

```

```
<simpleType name="sectionType">
  <restriction base="token">
    <pattern value="0[1-9][0-9]{3}" />
  </restriction>
</simpleType>
```

A.3.2 Definitions specific to the Data Download Service

These definitions are used by the Data Download Service.

```
<simpleType name="imsType">
  <restriction base="token">
    <maxLength value="253" />
    <pattern value="([A-Za-z0-9-]{1,31})(.[A-Za-z0-9-]{1,31}){1,63}" />
  </restriction>
</simpleType>

<simpleType name="pstnType">
  <restriction base="token">
    <pattern value="7[2-9][0-9]{6}" />
  </restriction>
</simpleType>
```

A.4 WSDL relating to faults

These definitions relate to the faults defined in ND1023 [3].

A.4.1 Fault elements

```
<element name="PolicyException" type="nd1024:PolicyExceptionType">
</element>

<complexType name="PolicyExceptionType">
  <sequence>
    <element name="messageId" type="string" />
    <element name="text" type="string" />
    <element name="variables" type="string" minOccurs="0"
      maxOccurs="unbounded" />
  </sequence>
</complexType>

<element name="ServiceException" type="nd1024:ServiceExceptionType">
</element>

<complexType name="ServiceExceptionType">
  <sequence>
    <element name="messageId" type="string" />
    <element name="text" type="string" />
    <element name="variables" type="string" minOccurs="0"
      maxOccurs="unbounded" />
  </sequence>
</complexType>
```

A.4.2 Fault messages

```
<message name="PolicyExceptionMessage">
  <part name="PolicyException" element="nd1024:policyException">
</message>
<message name="ServiceExceptionMessage">
  <part name="ServiceException" element="nd1024:serviceException">
</message>
```

A.4.3 Fault operations

```
<fault name="PolicyException" message="nd1024:PolicyExceptionMessage" />
<fault name="ServiceException" message="nd1024:ServiceExceptionMessage" />
```

Annex B (informative): Example messages

These are examples of messages sent as part of the interfaces in this specification.

B.1 NotifyRequest message

```
<notificationRequest ID="example-123456">  
  <notify Section="01414" Checkpoint="0805131241" />  
  <notify Section="02890" Checkpoint="0805131241" />  
</notificationRequest>
```

B.2 NotifyResponse message

```
<notificationResponse ID="example-123456" Count="2" />
```

B.3 DownloadRequest message

Example requesting two incremental downloads and two full downloads:

```
<downloadRequest ID="example-123457">
  <download Section="01414" Type="incremental" Checkpoint="0805131241" />
  <download Section="01632" Type="full" />
  <download Section="02890" Type="incremental" Checkpoint="0805131241" />
  <download Section="08456" Type="full" />
</downloadRequest>
```

B.4 DownloadResponse message

```
<downloadResponse ID="example-123457">
  <sectionData Section="01414" Type="incremental" Checkpoint="0805131250">
    <entry first="960223" ttl="720">
      <pstn>73100244</pstn>
    </entry>
    <entry first="960224" last="960299" ttl="720">
      <pstn>74016285</pstn>
      <ims>cs3.p28.dg.example</ims>
    </entry>
    <entry first="960100" last="960199" ttl="720" />
    <entry first="9600" last="9603" ttl="720">
      <send-n>l2</send-n>
    </entry>
    <ecc Algorithm="3:SHA1" Value="d463675b6022c7c1cecf2ad2e55a5caa538284d4" />
    <fullEcc Algorithm="3:SHA1" Checkpoint=""
      Value="3048c724b10a234e8358dff23486281c3494ee71" />
  </sectionData>
  <sectionData Section="01632" Type="full">
    <entry first="000000" last="999999" ttl="720" />
    <ecc Algorithm="0:SHA1" Value="ae9a8d1954072074f8eaa9183ae3e836f5405820" />
    <ecc Algorithm="3:SHA1" Value="b32e177d350d48e7d029a590e696ba2b7b81d35e" />
  </sectionData>
  <sectionData Section="02890" Type="incremental" Checkpoint="0805131249">
    <entry first="960954" ttl="720" />
    <ecc Algorithm="3:SHA1" Value="4cad9ba2af6661ffeb364f09766d5130be509412" />
  </sectionData>
  <sectionData Section="08456" Type="omitted" Checkpoint="0805131249" />
</downloadResponse>
```

Annex C (informative): ECC rationale

This annex explains the rationale behind the choices made in designing the ECC algorithm in section 9.

Canonicalisation: the ECC could check the *XML* or the *data* transmitted. The advantage of checking the *XML* is that no additional processing is required before doing the check. On the other hand, the *XML* representing a given dataset can have many forms – for example, the white space can be altered, some items can be reordered, and elements with no contents can be written as `<XXX />` or as `<XXX></XXX>`. This means that it is only possible to check the raw *XML* and not any data derived from it. In particular, there is no easy way to define the checksum of the entire Section, since there are an infinite number of different *XML* documents that could represent it (apart from the cosmetic changes mentioned earlier, several numbers could be grouped into a single `<entry>` element or left separate).

Checking the data does require the conversion of the data to some canonical form. This could be a form of *XML*. However, there is no obvious benefit (this particular form is unlikely to be the one chosen for transmission) and so a minimal format containing only the data was chosen.

The use of spaces to separate fields was an arbitrary choice based on the fact that no field will contain spaces. The rule that trailing empty fields and their spaces are omitted allows further fields to be added in the future.

Batching: if the ECC was calculated directly over the entire data, this could be CPU-intensive. A full Section can contain 1,111,111 separate entries. This is going to be 20MB to 50MB long. Requiring a complete recalculation of the ECC for every single change would therefore be onerous. Instead, the data is divided into batches – at batching level 3 there are up to 1001 batches, each containing at most 1,111 entries and therefore being at most 40kB to 50kB in size. A single change to the data will then only involve re-computing the ECC for one batch and then repeating the Consolidation stage, which is explicitly designed to be efficient.

Note that a batching level of 0 is equivalent to calculating the ECC by simply hashing the entire data.

The lines in the data could have been ordered either lexicographically or numerically. The former was chosen for two reasons. Firstly, it eliminates questions about the relative order of 123 and 0123. Secondly, numerical ordering means that SEND-N records will end up in separate batches to records holding destination groups. Adding or removing the number (say) 960456 can potentially affect the SEND-N records for 9, 96, 960, 9604, and 96045. Lexicographical ordering means that the changed records will all be in the same batch or, at worse, in two batches (the one holding the actual number and the additional batch containing those records shorter than the batching level), while numerical ordering could put all 6 records in different batches.

Consolidation: this stage has been designed to be fast. The use of XOR also introduces efficiencies in recalculating the ECC of the full Section after a change to the data. An implementation need only store the current full ECC and the ECC of each batch. Then, if a record changes, it executes the following procedure:

```
Temp := Previous_ECC XOR Batch_ECC [affected_batch];
recalculate Batch_ECC [affected_batch];
New_ECC := Temp XOR Batch_ECC [affected_batch];
```

Similarly, if two batches are affected:

```
Temp := Previous_ECC XOR Batch_ECC [affected_batch_1] XOR Batch_ECC [affected_batch_2];
recalculate Batch_ECC [affected_batch_1];
recalculate Batch_ECC [affected_batch_2];
New_ECC := Temp XOR Batch_ECC [affected_batch_1] XOR Batch_ECC [affected_batch_2];
```

The choice of batching level allows a tradeoff between CPU time to calculate individual batch ECCs and the number of batch ECCs that need to be stored.

History

Document history		
1.1.1	December 2008	Initial issue