

**THE CENTRAL BANK OF THE RUSSIAN FEDERATION
(BANK OF RUSSIA)**

**Unified Formats
of Electronic Banking Messages**

DIRECTORIES

Version 2018.3.0

Moscow

2018

Table of Contents

1. REFERENCES.....	3
2. DIRECTORIES AND CLASSIFIERS USED.....	4
3. TERMS, DEFINITIONS, AND ABBREVIATIONS	5
4. EPM ELEMENTS	8
5. ESIM ELEMENTS	9
6. EPM (ESIM) PACKET DETAILS	10
7. TECHNICAL INFORMATION ON UFEBM.....	11
Execution of an XML document.....	11
8. DESCRIPTION OF TRANSFORMATIONS OF AN XML DOCUMENT FOR CONVERTING IT TO A NORMALISED TYPE.....	12
A. Removal of processing instructions from an XML document	12
B. Removal of attributes from the namespace “http://www.w3.org/2001/XMLSchema-instance” from components of an XML document.....	13
C. Ordering namespace prefixes in all components of an XML document pursuant to the defined rule 14	
D. Removal of child textual nodes containing only white spaces from each component of an XML document	16
9. DESCRIPTION OF TRANSFORMATIONS OF AN XML DOCUMENT FOR CONVERTING IT TO A CANONICAL TYPE	18
10. DESCRIPTION OF BASE64 ENCODING ALGORITHM	22

1. References

The UFEBM Albums contain references to the following standards, regulations, and specifications:

[Regulation 383-P] Regulation on the Rules of Funds Transfer: Regulation of the Bank of Russia No. 383-P, dated 19 June 2012;

[Regulation 595-P] Regulation on the Payment System of the Bank of Russia: Regulation of the Bank of Russia No. 595-P, dated 06 July 2017;

GOST ISO 8601–2001 System of Standards Related to Information, Librarianship and Publishing. Date and Time Presentation. General Requirements;

Ordinance of the Bank of Russia No. 4257-U, dated 29 December 2016, 'On the List, Forms, Rules, and Procedure for the Preparation and Submission of Reporting Forms by Structural Business Units of the Bank of Russia to the Central Bank of the Russian Federation';

[XML] Extensible Markup Language (XML) 1.0 [Electronic resource]: W3C Recommendation 26 November 2008. – Fifth edition. – Access mode: <http://www.w3.org/TR/REC-xml>;

[XML-ns] Namespaces in XML [Electronic resource]: World Wide Web Consortium 14-January-1999. – Access mode: <http://www.w3.org/TR/REC-xml-names>;

[XML-schema] XML Schema Part 1: Structures [Electronic resource]: W3C Recommendation 2 May 2001. – First Edition. – Access mode: <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>; XML Schema Part 2: Datatypes [Electronic resource]: W3C Recommendation 2 May 2001. – First Edition. – Access mode: <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>;

[XML-c14n] Canonical XML Version 1.0 [Electronic resource]: W3C Recommendation 15 March 2001. – Access mode: <http://www.w3.org/TR/xml-c14n>;

[XPath] XML Path Language (XPath) Version 1.0 [Electronic resource]: W3C Recommendation 16 November 1999. – Access mode: <http://www.w3.org/TR/1999/REC-xpath-19991116>;

[base64] Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies [Electronic resource]: November 1996. – Access mode: <http://www.ietf.org/rfc/rfc2045.txt>;

[SOAP12] SOAP Version 1.2 Part 0: Primer [Electronic resource]: W3C Recommendation 24 June 2003. – Access mode: <http://www.w3.org/TR/soap12-part0/>; SOAP Version 1.2 Part 1: Messaging Framework [Electronic resource]: W3C Recommendation 24 June 2003. – Access mode: <http://www.w3.org/TR/soap12-part1/>; SOAP Version 1.2 Part 2: Adjuncts [Electronic resource]: W3C Recommendation 24 June 2003 – Access mode: <http://www.w3.org/TR/soap12-part2/>; SOAP Version 1.2 Specification Assertions and Test Collection [Electronic resource]: W3C Recommendation 24 June 2003. – Access mode: <http://www.w3.org/TR/2003/REC-soap12-testcollection-20030624/>;

[RFC2616] RFC 2616 Hypertext Transfer Protocol -- HTTP/1.1 [Electronic resource]: IETF RFC 2616 June 1999 – Access mode: <http://www.ietf.org/rfc/rfc2616.txt>;

[RFC2821] RFC 2821 Simple Mail Transfer Protocol [Electronic resource]: IETF RFC 2821 April 2001 – Access mode: <http://www.ietf.org/rfc/rfc2821.txt>;

[deflate] Request for Comments 1951. DEFLATE Compressed Data Format Specification version 1.3 [Electronic resource]: May 1996. – Access mode: <http://www.ietf.org/rfc/rfc1951>;

[zlib] Request for Comments 1950. ZLIB Compressed Data Format Specification version 3.3 [Electronic resource]: May 1996. – Access mode: <http://www.ietf.org/rfc/rfc1950>.

2. Directories and Classifiers Used

The designations of directories and classifiers below are used to describe application types.

[RF BIC] Directory of the Bank Identification Codes of Settlement Participants in the Territory of the Russian Federation (Directory of RF BICs). The Structure of the RCBIC, the Procedure for Its Assignment to the Participants of the Payment System and the Customers of the Bank of Russia That Are Not Participants of the Payment System, and the Elements of the Directory of Bank Identification Codes in the Payment System of the Bank of Russia: Annex 6 to the Regulation on the Payment System of the Bank of Russia: Regulation of the Bank of Russia No. 595-P, dated 06 July 2017.

[INN] Directory of the Taxpayer Identification Number. On Approval of the Procedure and Conditions of the Assignment, Application, and Change of a Taxpayer Identification Number and the Forms of Documents Used for Accounting of Legal Entities and Individuals in a Tax Authority: Order of the Ministry of Taxation No. GB-3-12/309, dated 27 November 1998.

[KPP] Directory of Tax Registration Reason Codes. On Approval of the Procedure and Conditions of the Assignment, Application, and Change of a Taxpayer Identification Number and the Forms of Documents Used for Accounting of Legal Entities and Individuals in a Tax Authority: Order of the Ministry of Taxation No. GB-3-12/309, dated 27 November 1998.

[UIMA] Unique Identifier of an Author of an Electronic Message.

[Document Code] Document Code. On Accounting Rules in the Central Bank of the Russian Federation (Bank of Russia): Regulation of the Bank of Russia No. 66-P, dated 1 January 2006, Annex 1; On Rules of Accounting in Credit Organisations Located in the Territory of the Russian Federation: Regulation of the Bank of Russia No. 302-P, dated 26 March 2007.

[Personal Account] Account Number. On Accounting Rules in the Central Bank of the Russian Federation (Bank of Russia): Regulation of the Bank of Russia No. 66-P, dated 1 January 2006, Annex 1; On Rules of Accounting in Credit Organisations Located in the Territory of the Russian Federation: Regulation of the Bank of Russia No. 302-P, dated 26 March 2007.

[Payment priority] Payment priority. Civil Code of the Russian Federation, Article 855, Clause 2;

[GOST ISO 8601–2001] GOST ISO 8601-2001 System of Standards Related to Information, Librarianship, and Publishing. Date and Time Presentation. General Requirements.

3. Terms, Definitions, and Abbreviations

The following terms and definitions are used in the UFEBM Albums:

Table 1. Terms and definitions

Term	Description
Automated System of Issue-Cash Works, AS ICW	An automated system designated for complex automation of issue-cash works in a regional division of the Bank of Russia and in subordinate main cash-settlement centres and cash-settlement centres.
AWS	Automated Workstation
AC Owner	A BoR BU, credit organization (branch), or other customer of the Bank of Russia whose AC is registered pursuant to the procedure established by a contract between the Bank of Russia and its customer
Globally Unique Identifier (GUID)	A 16-byte (128-bit) identifier that is guaranteed to be unique. The probability of the generation of two identical GUIDs is negligibly low
UC	User Code
SC	Security Code
CO	Credit organisation
CO/BoR Customer	Credit organisations and other customers of the Bank of Russia
Authentication Code; AC	Data used for certification of the right to dispose of funds (for EMs containing instructions in electronic form), integrity control, and authentication of EMs
MPP	Multicycle Payment Processing
CPP	Continuous Payment Processing
NPCS	National payment card system
Packet of Electronic Messages	One or several EMs (EPMs/ESIMs) with an established AC, where an AC is not established on each EM forming part of the packet
Software	Software programmes
Business Unit of the Bank of Russia; BoR BU	The Main Directorate of the Bank of Russia, a branch, a branch that is a national bank of the Main Directorate of the Bank of Russia, a structural business unit of the Central Office of the Bank of Russia, a cash-settlement centre, a cash centre, or a field division of the Bank of Russia (Regulation 595-P)
Centralised Component of the Payment System of the Bank of Russia; BoR PS	An automated system for the automation of settlements and creation of a common information environment for functional interaction between all banking participants (BoR BUs, commercial banks and their branches, non-credit organisations, and other customers of the Bank of Russia)
SABS	Specialised Automated Banking System
SVK	A unified transport environment of electronic interaction between regional divisions of the Bank of Russia and customers of the Bank of Russia
FMS	Financial Messaging System of the Bank of Russia
VAS CHC; VAS	Value Accounting System of the Cash Handling Centre of the Bank of Russia MD for Saint Petersburg
FMOD of the Bank of Russia; FMOD	Financial Market Operations Department of the Bank of Russia

Transport Protocol	A protocol implementing a network service, e.g., e-mail, web access, guaranteed delivery of messages (SMTP, HTTP, WMQ), and using a network-level protocol (e.g., TCP or UDP)
Transport message (e-mail message, HTTP message, WebSphere MQ message)	A combination of information components executed pursuant to the requirements of the transport protocol (SMTP, HTTP, IBM WebSphere MQ protocol, etc.) that is used in an exchange
BoR RD	Regional Division of the Bank of Russia
Participants of EM Exchange with the Bank of Russia; Participants	Credit organisations, branches of credit organisations, and other customers of the Bank of Russia that have entered into a contract on the exchange of electronic messages when making settlements through the settlement system of the Bank of Russia with the Bank of Russia (the 'Exchange Contract')
Direct Participant	Organisations meeting the following criteria: <ul style="list-style-type: none"> - the organisation is a credit organisation (its branch), the Federal Treasury or its territorial body, or another organisation that can be a direct participant pursuant to Part 7 of Article 21 of Federal Law No. 161-FZ, dated 27 June 2011; - the organisation has a bank (correspondent) account (sub-account) in the Bank of Russia pursuant to a bank (correspondent) account (sub-account) contract.
Indirect Participant	Organisations meeting the following criteria: <ul style="list-style-type: none"> - the organisation can be a payment system participant pursuant to Part 1 of Article 21 of Federal Law No. 161-FZ, dated 27 June 2011; - the organisation is a customer of a credit organisation (its branch) that is a direct participant; - the organisation is provided with access to funds transfer services using instructions in electronic form pursuant to the terms of the correspondent account (sub-account) contract entered into by the Bank of Russia and a credit organisation (its branch) that is a direct participant; - the organisation is not a customer of the Bank of Russia.
Main Liquidity Pool Participant	Owner of a bank (correspondent) account (sub-account) opened in the Bank of Russia which is the main account of a liquidity pool
Subordinate Liquidity Pool Participant	Owner of a bank (correspondent) account (sub-account) opened in the Bank of Russia which is a subordinate account of a liquidity pool
UFE BM	Unified Formats of Electronic Banking Messages
Formats of Electronic Messages	An ordered sequence of symbols included in an EM pursuant to uniform rules, presented in a formalised form, in the stipulated sequence and dimension, that is used for transmission through communication channels and its processing
AS ICW IPC	Information Processing Centre of the Automated System of Issue-Cash Works
MEC	Message Exchange Centre
Electronic Message; EM	A set of data corresponding to the electronic format established by the Bank of Russia, suitable for the unambiguous perception of its content, provided with an authentication code
Electronic Payment Message; EPM	An electronic message that is the basis for making transactions in respect of the accounts of credit organisations (branches) and other customers of the Bank of Russia opened in business units of the Bank of Russia, with ACs established, and having equal legal force with settlement documents

	on paper media signed by handwritten signatures of authorised persons and certified with the imprint of a seal
Electronic Service and Information Message; ESIM	An electronic message with an AC established securing information exchange when making settlements and transactions in respect of accounts opened in business units of the Bank of Russia (requests, reports, statements of accounts, documents connected with the extension of loans by the Bank of Russia, etc.)
Electronic Messages; EM	Funds transfer instructions and electronic service and information messages
CLS Bank International; CLS Bank	The settlement bank of the CLS system
HTTP (Hypertext Transfer Protocol)	Hypertext transfer protocol [RFC2616]
RTGS (Real Time Gross Settlement)	Real time gross settlements
Schema	A logical and physical definition of data components, physical characteristics, and internal relations
SMTP (Simple Mail Transfer Protocol)	Simple mail transfer protocol [RFC2821]
S.W.I.F.T. (Society of Worldwide Interbank Financial Telecommunications); SWIFT	Society of Worldwide Interbank Financial Telecommunications
W3C (World Wide Web Consortium)	World Wide Web Consortium, Internet consortium
WMQ	The system of message transfer with guaranteed delivery WebSphere MQ (earlier, MQSeries). This abbreviation is also used to designate the transport protocol of the guaranteed delivery of messages implemented in the WebSphere MQ system
XML (extensible markup language)	Extensible (public) markup language
XML schema	Document structure description language. Provides for the description of the admissible structure of a document and, possibly, data types in attribute values and component contents. At the present, there are several such languages
XML document	A set of data corresponding to the requirements [XML]
XML namespaces	Method for the clarification (qualification) of names of components and attributes by comparing a set of names of a particular sphere of applicability technically expressed in the form of the universal resource identifier (URI)

4. EPM Elements

The composition and the procedure of filling the fields (elements) of an EPM shall correspond to the composition and the procedure of filling the fields (elements) of settlement documents on paper media.

An exception is provided for the following elements:

- a name and a registered address of the paying bank and the receiving bank (fields 10 and 13) are not indicated in the EPM (in copies of the EPM on paper, these elements are to be filled in pursuant to the Directory of RF BICs);
- the elements 'Payment due date' and 'Payment purpose' (fields 19, 20, respectively) are not used in the EPM before the receipt of instructions of the Bank of Russia;
- the amount in words (field 6) is not indicated in EPM (in EPM copies on paper, the element is generated based on software based on the digital value of the amount);
- the document name (field 1), the form number under OKUD (National Index of Administrative Documents) (field 2), and graphical elements (stamps, signatures, and imprints of a seal to be placed in fields 43, 44, 45, 46, 47, 69) are not indicated in the EPM;
- in the field 'date of submission of documents to the recipient's bank' (field 48), the date value contained in the field of the settlement document 'notes of the recipient's bank' is to be indicated (field 48).

In addition to elements contained in settlement documents on paper media, the EPM contains elements to be filled in pursuant to the following procedure.

The payment order and the payment instruction in the form of an EPM in cases when they are made on the basis of an EPM (partial payment, crediting funds for the intended purposes after visual control of the EPM, return of funds under an EPM, etc.) contain a reference group of elements allowing to unambiguously identify the initial EPM on the basis of which this EPM is generated and comprising 'Sequential number of the initial EPM,' 'Initial EPM composition date,' and 'Unique identifier of the initial EPM author' (fields 203, 204, 205, respectively).

In the case of partial acceptance, a payment request in the form of an EPM—in addition to the accepted amount payable—contains the amount of the initial settlement document presented for acceptance (field 150).

Table 2. EPM element groups

Group	Elements	Note
Digit-based group	Sequential number of the EM (unique for each participant or BoR BU during a day); EM composition date; Unique identifier of the EM author.	Always included. The element "Unique identifier of the EM author" for EMs originating from the BoR PS is to be filled in with the BoR BU's UIMA.
Reference group	Sequential number of the initial EM; Initial EM composition date; Unique identifier of the initial EM author.	A payment order and a payment instruction in the form of an EPM, when they are made on the basis of an EPM (partial payment, return of funds under EPM, etc.), shall contain a reference group of elements making it possible to unambiguously identify the initial EPM on the basis of which this EPM was composed.

5. ESIM Elements

Along with the EPM, the participant, when making cashless payments with the payment system of the Bank of Russia, exchanges **ESIMs** with its servicing BoR BU pursuant to the procedure and formats stipulated by the current document and by the Exchange Contract.

Table 3. ESIM element groups

Group	Elements	Note
Digit-based group	Sequential number of the EM; EM composition date; Unique identifier of the EM author.	Always included. The element "Unique identifier of the EM author" for EMs originating from the BoR PS is to be filled in with the BoR BU's UIMA.
Additional information	Unique identifier of the EM recipient. EM composition time.	Application information. Not used in the transport system of the Bank of Russia.
Reference group	Sequential number of the initial EM; Initial EM composition date; Unique identifier of the initial EM author.	To be filled in pursuant to the values of the elements of the digit-based group of the EPM or ESIM in response to which or with a query about which this ESIM is being sent
Informative part	Result code; Information on transactions made; Explanation text, etc.	Established by technical regulations of the Bank of Russia
Additional elements		Optional for filling in

6. EPM (ESIM) Packet Details

EPM (ESIM) packet shall mean one or several EPMs (ESIMs) signed with an AC (the procedure is to be determined by regulations of the Bank of Russia), where each EPM (ESIM) forming part of the packet is not signed with an AC. The EM packet has a common group of elements that relates to the entire set of EPMs forming part of the packet.

The Bank of Russia has the right to establish a restriction in the Exchange Contract on the total number of EPMs forming part of an EPM packet and its composition.

The EPM packet has a common group of elements that relates to the aggregate of EPMs forming part of the packet and comprises 'Total number of EPMs forming part of the packet' and '"Total amount of EPMs forming part of the packet'

7. Technical Information on UFEBM

UFEBM were developed pursuant to the following specifications:

- "Extensible Markup Language (XML) 1.0" [XML];
- "Namespaces in XML" [XML-ns];
- "XML Schema Part 1: Structures" and "XML Schema Part 2: Datatypes" [XML-schema].

Execution of an XML document

Electronic messages are to be drawn up pursuant to [XML]. XML documents are to be transmitted in the encoding WINDOWS-1251 or UTF-8.

XML documents shall start with an XML declaration that defines the XML version and the applicable encoding pursuant to [XML].

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

Note: An XML declaration specifying the XML version in an XML document is mandatory. If the applicable encoding is not specified, an XML document is considered to be in UTF-8 encoding.

For XML documents, the described namespaces and their prefixes are to be declared (namespaces can be used by default). For components containing the value of an AC or SC, the necessary namespaces can be determined in the component itself.

In an XML document, the belonging of all components to a particular namespace shall be set (by setting the namespace by default or explicitly with the use of prefixes). Attributes in an XML document are drawn up as so-called non-classified local components—that is, attributes are defined forming part of the component and do not relate to any designated namespace (a prefix determining a designated namespace does not apply to the attribute). In the schema, the belonging of components and attributes to a particular namespace is set globally with the following values:

```
elementFormDefault="qualified" attributeFormDefault="unqualified".
```

Note: A namespace prefix does not have any meaning and is used only for linking the names of components and attributes to the namespace.

It is prohibited to use the following structures that are not explicitly described in the current document when generating XML messages:

- comments,
- processing instructions,
- document type declarations (DTD),
- definitions of namespaces that differ from the described ones,
- attributes from the namespace "http://www.w3.org/2001/XMLSchema-instance"¹,
- attributes from the namespace "http://www.w3.org/XML/1998/namespace",
- CDATA sections.

¹ The presence of the attributes schemaLocation or noNamespaceSchemaLocation from the namespace "http://www.w3.org/2001/XMLSchema-instance" specifying the schema location (as a prompt for the processing programme) is not permitted.

8. Description of Transformations of an XML Document for Converting It to a Normalised Type

This annex describes transformations of an XML document for converting it to a normalised type (normalisation). These transformations apply to an XML document containing the protected data of an EM (EM packet) and are performed before canonicalisation of an XML document containing the protected data of the EM (EM packet).

Identifier of the algorithm describing the transformations:

“urn:cbr-ru:dsig:v1.1#normalization”.

When describing the algorithm of normalisation of an XML document for addressing parts of an XML document containing the protected data of an EM (EM packet), the language [XPath] is used.

XPath represents an XML document in the form of a node tree. Nodes can be of different types: component nodes, attribute nodes, namespace nodes, processing instruction nodes, comment nodes, textual nodes. When performing an electronic exchange in the cashless payment system, it is prohibited to use the following nodes in XML documents:

- comment nodes,
- processing instruction nodes,
- namespace nodes that differ from the described ones,
- attribute nodes from the namespace "http://www.w3.org/2001/XMLSchema-instance",
- textual nodes, if the parent node contains child nodes of components.

Thus, when processing and storing information from an XML document, only a part of the nodes are used, and all other nodes are ignored. Information that is not used when performing the electronic exchange shall not be protected with SCs; therefore, prior to generating the SC, it is necessary to apply transformations that remove excessive information from an XML document (unused nodes). This allows to protect information with no regard for special features of markup.

Below is the minimum list of the necessary transformations of an XML document for converting it to a normalised type :

- removal of processing instructions from an XML document;
- removal of attributes from the namespace "http://www.w3.org/2001/XMLSchema-instance" from components of an XML document;
- sorting namespace prefixes in all components of an XML document pursuant to the defined rule;
- removal of child textual nodes containing only white spaces from each component of an XML document.

The order of transformations matters and shall be performed exactly as described.

A. Removal of processing instructions from an XML document

Use of processing instructions is prohibited in UFEBM of the Bank of Russia; therefore, they shall be ignored when accepting EMs and are not protected with SCs.

A processing instruction shall have the following syntax:

```
<?addressee[_contents]>>
```

where

- addressee shall mean an address used for identification of the application for which this instruction is designated;
- contents shall mean the contents of a processing instruction, including an addressee and any white spaces immediately following it;

- an optional syntactic unit is presented in square brackets;
- for designation of a space character, the underscore character is used (_).

For each processing instruction, except for processing instructions placed in the document type declaration, a respective node is created.

Algorithm for removal of processing instructions from an XML document:

For each node of a tree representing an XML document containing the protected data of an EM (EM packet), the following transformation is performed:

- if this node is a processing instruction node, it is removed.

Example

XML document prior to transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<?xml-stylesheet type="text/xsl" href="UniESID1.xslt"?>
<ED202 xmlns="urn:cbr-ru:ed:v2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:cbr-ru:ed:v2.0 UniDoc1_1.xsd"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1">
  <EDRefID EDNo="7" EDDate="2003-04-14" EDAuthor="4525545000"/>
</ED202>
```

XML document after transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<ED202 xmlns="urn:cbr-ru:ed:v2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:cbr-ru:ed:v2.0 UniDoc1_1.xsd"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1">
  <EDRefID EDNo="7" EDDate="2003-04-14" EDAuthor="4525545000"/>
</ED202>
```

The processing instruction <?xml-stylesheet type="text/xsl" href="UniESID1.xslt"?> has been removed from an XML document. The XML declaration <?xml version="1.0" encoding="Windows-1251"?> is not a processing instruction; therefore, it has been left in an XML document.

B. Removal of attributes from the namespace

“http://www.w3.org/2001/XMLSchema-instance” from components of an XML document

Use of attributes from the namespace "http://www.w3.org/2001/XMLSchema-instance" is prohibited in UFEBMs of the Bank of Russia. Attributes from this namespace shall be ignored when processing EMs and are not protected with SC.

The algorithm for removal of attributes from the namespace "http://www.w3.org/2001/XMLSchema-instance" from components of an XML document:

For each node of a tree representing an XML document containing the protected data of an EM (EM packet), the following transformation is to be performed:

a) if this node is a component node, the following transformation shall be performed for each of its child nodes:

1) if this child node is an attribute node, and this attribute belongs to the namespace "http://www.w3.org/2001/XMLSchema-instance," and the local name of the attribute coincides with one of the following identifiers: schemaLocation, noNamespaceSchemaLocation, type or nil, it is to be removed.

Example

XML document prior to transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<ED202 xmlns="urn:cbr-ru:ed:v2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:cbr-ru:ed:v2.0 UniDoc1_1.xsd"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
```

```
EDReceiver="4525000000" InquiryCode="1">
  <EDRefID EDNo="7" EDDate="2003-04-14" EDAuthor="4525545000"/>
</ED202>
```

XML document after transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<ED202 xmlns="urn:cbr-ru:ed:v2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1">
  <EDRefID EDNo="7" EDDate="2003-04-14" EDAuthor="4525545000"/>
</ED202>
```

The attribute `xsi:schemaLocation` from the namespace "`http://www.w3.org/2001/XMLSchema-instance`" has been removed from the component ED202.

C. Ordering namespace prefixes in all components of an XML document pursuant to the defined rule

In UFEBM of the Bank of Russia, the value of a prefix of a namespace does not matter and can be any value. To exclude the necessity of keeping information on special features of markup, rules for ordering and converting namespace prefixes in all components of an XML document containing the protected data of an EM (EM packet) to a particular type are set.

Transformation converts an XML document to a type satisfying the following requirements:

- The namespace prefixes in use are set in the form `n1`, `n2` etc. The positive integer following `n` is called a prefix index.
- Namespaces are never inherited.
- Namespaces are never applied by default.

Only such namespaces to which a component and its attributes belong shall be declared in a component. Declarations of other namespaces shall be **deleted** to prevent a conflict of names.

Note: This approach has been chosen to secure context-insensitive representation: a component's form does not depend on where it occurs in an XML document.

Algorithm for ordering namespace prefixes in all components of an XML document pursuant to the defined rule:

For each node of a tree component representing an XML document containing the protected data of an EM (EM packet), the following transformation is to be performed:

- a) the list of namespaces is created (URI);
 - 1) the list components are the namespaces to which an element and its attribute belong;
 - 2) the list of namespaces is sorted in lexicographic order, and all duplicating namespaces are deleted;
 - 3) pursuant to the sorted order, each namespace from the list, a prefix `n1`, `n2`, ..., `nXX` is assigned (prefix indices are always set by consecutive integers, starting from 1);
- b) a part of an element name representing a namespace prefix is brought into conformity with the list of namespaces;
- c) for each child node of this component node, the following transformation is performed:
 - 1) if this child node is a namespace node, it is removed;
 - 2) if this child node is an attribute node, the part of the attribute name representing a namespace prefix is brought into conformity with the list of namespaces.
- d) child nodes of namespaces are added to the element node pursuant to the list of namespaces.

Note: In each component, all namespaces to which this component and its attributes belong are declared. This approach has been chosen to simplify the algorithm.

Examples

1 Ordering namespace prefixes in an XML document

XML document prior to transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<ED202 xmlns="urn:cbr-ru:ed:v2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1">
  <EDRefID EDNo="7" EDDate="2003-04-14" EDAuthor="4525545000"/>
</ED202>
```

XML document after transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<n1:ED202 xmlns:n1="urn:cbr-ru:ed:v2.0"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1">
  <n1:EDRefID xmlns:n1="urn:cbr-ru:ed:v2.0" EDNo="7" EDDate="2003-04-14"
  EDAuthor="4525545000"/>
</n1:ED202>
```

The element **ED202** belongs to the namespace **"urn:cbr-ru:ed:v2.0"** declared as the namespace by default; its attributes belong to a non-specified namespace. Therefore, the list of namespaces consists of one component **"urn:cbr-ru:ed:v2.0"**, the prefix **n1** is assigned to it. This namespace is to be declared in the element; the prefix **n1** is assigned to it. The prefix **n1** is placed before the element name **ED202**, which belongs to the namespace **"urn:cbr-ru:ed:v2.0"**.

The element **EDRefID** belongs to the namespace **"urn:cbr-ru:ed:v2.0"** declared as the namespace by default in the parent element **ED202**; its attributes belong to a non-specified namespace. Therefore, the list of designations of namespaces consists of one component **"urn:cbr-ru:ed:v2.0"**, the prefix **n1** is assigned to it. This namespace is declared in the element; the prefix **n1** is assigned to it. The prefix **n1** is placed before the element name **EDRefID**, which belongs to the namespace **"urn:cbr-ru:ed:v2.0"**.

The declaration of the namespace **"http://www.w3.org/2001/XMLSchema-instance"** has been removed as neither the component nor its attributes belong to this namespace.

This example is sufficient for understanding the procedure of ordering namespace prefixes within **UFE BM**. As namespaces are not used by default, and only namespaces to which an element and its attributes belong to are declared, and attributes in an XML document are defined as part of the element, the list of namespaces always consists of one element to which the prefix **n1** is assigned. Thus, within **UFE BM**, the procedure of ordering namespace prefixes in all elements is reduced to declaring a single namespace with the prefix **n1** in each element and placing this prefix before the element name.

2 Ordering namespace prefixes in an abstract XML document

XML document before transformation (lines are numbered for the convenience of commenting):

```
1 <x:doc xmlns:x="http://w3.org/2"
2     xmlns:y="http://w3.org/1"
3     xmlns:z="http://w3.org/3"
4     xmlns:w="http://w3.org/4"
5     xmlns:n1="http://w2.org/1">
6   <x:e a="a"/>
7   <x:e x:a="x:a"/>
8   <e x:a="x:a"/>
9   <e x:a="x:a" a="a" xmlns="http://w2.org/2"/>
10  <e x:a="x:a" y:a="y:a"/>
11  <e x:a="x:a" x:b="x:b"/>
12  <x:e x:a="x:a" x:b="x:b" xmlns:v="http://w3.org/6"
  xmlns:z="http://w3.org/5">
13    <z:e x:a="x:a" x:b="x:b" n1:c="n1:c"/>
14  </x:e>
15 </x:doc>
```

XML document after transformation:

```
1 <n1:doc xmlns:n1="http://w3.org/2">
6   <n1:e a="a" xmlns:n1="http://w3.org/2"/>
7   <n1:e n1:a="x:a" xmlns:n1="http://w3.org/2"/>
8   <e n1:a="x:a" xmlns:n1="http://w3.org/2"/>
9   <n1:e a="a" n2:a="x:a" xmlns:n1="http://w2.org/2"
  xmlns:n2="http://w3.org/2"/>
10  <e n1:a="y:a" n2:a="x:a" xmlns:n1="http://w3.org/1"
  xmlns:n2="http://w3.org/2"/>
11  <e n1:a="x:a" n1:b="x:b" xmlns:n1="http://w3.org/2"/>
```

```

12      <n1:e n1:a="x:a" n1:b="x:b" xmlns:n1="http://w3.org/2">
13      <n3:e n1:c="n1:c" n2:a="x:a" n2:b="x:b"
xmlns:n1="http://w2.org/1" xmlns:n2="http://w3.org/2"
xmlns:n3="http://w3.org/5"/>
14      </n1:e>
15  </n1:doc>

```

Таблица Table 3 demonstrates the procedure for ordering namespace prefixes for this example.

Namespaces "http://w3.org/1", "http://w3.org/3", "http://w3.org/4", and "http://w2.org/1" declared in the element x:doc (lines 1–5) are not subject to the rule of ordering namespace prefixes (the element does not belong to these namespaces) and are removed to avoid a conflict of names (after ordering the prefix n1 that has already been assigned to the namespace "http://w2.org/1" will be assigned to the namespace "http://w3.org/2").

The namespaces "http://w3.org/6" and "http://w3.org/5" declared in the element x:e (line 12) are not subject to the rule of collating namespace prefixes and are removed.

T a b l e 4. Ordering namespace prefixes in all elements of an abstract example

Lines	Element and its attributes before and after ordering prefixes			Sorted list of namespaces with the assigned prefixes	
	Name before	Namespace to which a element/attribute belongs	Name after	Namespace	Prefix
1–5, 15	doc	"http://w3.org/2"*	n1:doc	"http://w3.org/2"	n1
6	x:e	"http://w3.org/2"*	n1:e	"http://w3.org/2"	n1
	a	not specified	a		
7	x:e	"http://w3.org/2"*	n1:e	"http://w3.org/2"	n1
	x:a	"http://w3.org/2"*	n1:a		
8	e	not specified	e	"http://w3.org/2"	n1
	x:a	"http://w3.org/2"*	n1:a		
9	e	"http://w2.org/2"**	n1:e	"http://w2.org/2"	n1
	a	not specified	a	"http://w3.org/2"	n2
	x:a	"http://w3.org/2"*	n2:a		
10	e	not specified	e	"http://w3.org/1"	n1
	x:a	"http://w3.org/2"*	n2:a	"http://w3.org/2"	n2
	y:a	"http://w3.org/1"*	n1:a		
11	e	not specified	e	"http://w3.org/2"	n1
	x:a	"http://w3.org/2"*	n1:a		
	x:b	"http://w3.org/2"*	n1:b		
12, 14	x:e	"http://w3.org/2"*	n1:e	"http://w3.org/2"	n1
	x:a	"http://w3.org/2"*	n1:a		
	x:b	"http://w3.org/2"*	n1:b		
13	z:e	"http://w3.org/5"***	n3:e	"http://w2.org/1"	n1
	x:a	"http://w3.org/2"*	n2:a	"http://w3.org/2"	n2
	x:b	"http://w3.org/2"*	n2:b	"http://w3.org/5"	n3
	n1:c	"http://w2.org/1"*	n1:c		
* The namespace has been declared in the parent element doc. ** The namespace has been declared by default. *** The namespace has been declared in the parent element x:e.					

D. Removal of child textual nodes containing only white spaces from each element of an XML document

Symbol data is grouped in text nodes. As much symbol data as possible is placed in each of these nodes: no other textual node having the same parent can come immediately before or after a textual node. The string value of a textual node is the same textual data. The textual data always contains at least one data symbol.

The use of textual nodes for information transfer in components in UFEBM of the Bank of Russia is permitted only in some leaf components (not containing child components), whereas transmission of contents containing only white spaces is prohibited in such components. All other textual nodes may not contain meaningful information (that is controlled by the XML schema). If textual nodes are used when formatting an XML document to improve legibility, they are ignored when processing EMs and are not protected with SCs.

Algorithm for removal of child textual nodes containing only white spaces from elements of an XML document:

For each node of a tree representing an XML document containing the protected data of an EM (EM packet), the following transformation is to be performed:

- a) if this node is a textual node that contains only white spaces, it is removed.

Notes

- 1 A white space consists of one or several of the symbols space (#x20), carriage return (#xD), string end (#xA), or tabulator (#x9). White spaces can serve to emphasise markup for better legibility.
- 2 Textual nodes containing at least one non white space symbol shall not be deleted.

Example

XML document prior to transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<n1:ED202 xmlns:n1="urn:cbr-ru:ed:v2.0"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1">
  <n1:EDRefID xmlns:n1="urn:cbr-ru:ed:v2.0" EDNo="7" EDDate="2003-04-14"
  EDAuthor="4525545000"/>
</n1:ED202>
```

XMI document after transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<n1:ED202 xmlns:n1="urn:cbr-ru:ed:v2.0"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1"><n1:EDRefID xmlns:n1="urn:cbr-
ru:ed:v2.0" EDNo="7" EDDate="2003-04-14" EDAuthor="4525545000"/></n1:ED202>
```

Normalisation of white spaces inside of tags depends on the parser that processed an XML document. In this example, white spaces inside the open tag <n1:ED202> are left unchanged.

9. Description of Transformations of an XML Document for Converting it to a Canonical Type

This annex describes transformations of an XML document for converting it to a canonical type (canonicalisation). These transformations apply to an XML document containing the protected EM (EM packet) data and are performed after normalisation of the XML document containing protected EM (EM packet) data.

Identifier of the algorithm describing the transformations:

“<http://www.w3.org/2001/10/xml-c14n>”.

When describing the algorithm for canonicalisation of an XML document for addressing the parts of an XML document containing the protected EM (EM packet) data, the language [XPath] is used.

The XML document containing the protected EM (EM packet) data must be canonicalised after applying the normalisation procedure to it so that its binary representation does not depend on the particular parser that processed the XML document or the operating system.

Standard canonicalisation [XML-c14n] determines the list of changes in an XML document after application of which the logical equivalence of the XML document with another XML document in canonical form can be established.

Comments in UFEBM are prohibited and shall not be protected with SCs. In connection with this, in UFEBM, canonicalisation without comments is used.

The list of changes in an XML document during canonicalisation without comments:

- an XML document is provided in UTF-8 encoding;
- line feed symbols are normalised in #xA; before parsing the XML document;
- attribute values are normalised;
- references to symbols and parsed entities are replaced with the respective text;
- CDATA sections are replaced with symbols contained therein;
- XML declarations `<?xml >` and document type declarations (DTD) are removed;
- empty components are transformed into a pair of open/closed tags (an element of the type `<Name/>` into the type `<Name><Name/>`);
- empty spaces outside a root component of an XML document and inside open and closed tags are normalised;
- all white spaces in the symbol contents (except for symbols removed during normalisation of line feed) are left in place;
- restrictive symbols of attributes are replaced with double quotes;
- special symbols in attribute values and symbol contents are replaced with references to symbols;
- excessive namespace declarations are removed from each element (a namespace declaration is considered to be excessive if the nearest parent element in the canonical form has the equivalent declaration of namespaces in its scope).
- attributes by default described in DTD are added to each element;
- the lexicographical order for namespace and attribute declarations is established inside each element (attributes are ordered by namespaces and local names);
- comments are removed.

Algorithm for converting an XML document containing the protected EM (EM packet) data to a canonical type:

Note: This algorithm is provided for an XML document in general but not for a subset of its nodes. Special features of the algorithm in respect of a subset of nodes (document subset) are presented in the original specification [XML-c14n].

The algorithm operates with the node set [XPath], completely representing an XML document to be canonicalised. When preparing a node set, the XML document processor (parser) must perform the following tasks:

- normalisation of string ends pursuant to the specification [XML];
- normalisation of attribute values pursuant to the specification [XML];
- replacement of CDATA sections with symbols contained therein pursuant to the specification [XML];
- resolution of symbol references and references to parsed entities pursuant to the specification [XML].

The node set [XPath] prepared by a parser is transformed into a byte array by generating of the respective UCS symbols for each node in the set in the order of occurrence in the XML document. After that, the result is encoded in UTF-8. Each node is processed only once. Element node processing includes processing of all child nodes for which this element node is an ancestor. Thus, after generation of the canonical form for the element, the node of this element and all nodes for which this node is an ancestor are removed from the node set. The element node is not removed from the node set until all its child nodes have been processed.

The text generated for each node depends on the node type.

Root node. The processing result is composed of the results of processing each child node in the order of occurrence in the XML document. The canonical form does not include XML declaration or document type declaration.

Element node. As the result of processing, the following text is generated: open angle bracket (<), qualified name of the element, result of processing of namespace nodes related to the current element node, result of processing of attribute nodes of this element, closed angle bracket (>), result of processing of child nodes of elements in the order of occurrence in the XML document, open angle bracket (<), forward slash (/), qualified name of the element and closed angle bracket (>).

- Nodes of namespaces related to the current element node are processed in lexicographical order. If the first namespace node in the list is not a namespace node by default (a namespace node by default has an empty URI and an empty local name but not empty contents), and the nearest ancestor element of the current element node contains a namespace node by default, the result of processing the namespace nodes of the element follows this text: blank, attribute cancelling declaration of the namespace by default (xmlns="").

- Attribute nodes are processed in lexicographical order.

Namespace node. The namespace node is ignored if the nearest ancestor element of the parent component of the node has a namespace node with the same local name and value as the current namespace node. Otherwise, the namespace node is processed like an attribute node, except for assignment of the local name xmlns to the namespace node by default, if any exists (the namespace node by default has an empty URI and an empty local name).

Attribute node. As the result of processing, the following text is generated: space, qualified name of an attribute, equal sign, open double quotes, modified string values, and closed double quotes. The modified string value represents the string value of an attribute normalised pursuant to the specification [XML]. The following symbols of the modified string value are replaced with references to symbols:

- ampersand (&) -> &
- open angle bracket (<) -> <
- double quotes (") -> "
- white spaces (#x9, #xA, and #xD) -> symbol references (symbol references are presented in hexadecimal format, in upper case, without leading zeroes).

Textual node. As the result of processing, a text is generated representing the string value of the text node, the string ends of which are normalised pursuant to the specification [XML]. The following symbols of the string value are replaced with references to symbols:

- ampersand (&) -> &
- open angle bracket (<) -> <
- closed angle bracket (>) -> >

- carriage return symbol (#xD) -> 

Processing instruction node. As the result of processing, the following text is generated: open symbols of processing instruction (<?), name of the addressee of the processing instruction, leading blank, string value, if not empty, and closed symbols of the processing instruction (?>). If the string value is empty, the leading blank is not to be added.

Note: As the result of normalisation, the node set of an XML document containing an EM (EM packet) should not contain processing instruction nodes.

Comment nodes are ignored in canonicalisation without comments.

Examples

1 Canonicalisation of an XML document

XML document prior to transformation:

```
<?xml version="1.0" encoding="Windows-1251"?>
<n1:ED202 xmlns:n1="urn:cbr-ru:ed:v2.0"
  EDNo="8" EDDate="2003-04-14" EDAuthor="4525545000"
  EDReceiver="4525000000" InquiryCode="1"><n1:EDRefID xmlns:n1="urn:cbr-
ru:ed:v2.0" EDNo="7" EDDate="2003-04-14" EDAuthor="4525545000"/></n1:ED202>
```

XML document after transformation:

```
<n1:ED202 xmlns:n1="urn:cbr-ru:ed:v2.0" EDAuthor="4525545000" EDDate="2003-04-
14" EDNo="8" EDReceiver="4525000000" InquiryCode="1"><n1:EDRefID
EDAuthor="4525545000" EDDate="2003-04-14" EDNo="7"></n1:EDRefID></n1:ED202>
```

After transformation in an XML document:

- the XML declaration <?xml version="1.0" encoding="Windows-1251"?> has been removed;
- empty spaces inside the open tag <n1:ED202> have been normalised;
- the excessive namespace declaration "urn:cbr-ru:ed:v2.0" in the element n1:EDRefID has been removed;
- the empty element n1:EDRefID has been transformed into a pair of open/closed tags: <n1:EDRefID></n1:EDRefID>;
- lexicographical order for namespace and attribute declarations has been established inside element n1:ED202 and n1:EDRefID.

2 Canonicalisation of an abstract XML document

XML document before transformation (lines are numbered for the convenience of commenting; formatting of the XML document is used for improved readability):

```
1 <n1:doc xmlns:n1="http://w3.org/2">
2   <n1:e a="a" xmlns:n1="http://w3.org/2"/>
3   <n1:e n1:a="x:a" xmlns:n1="http://w3.org/2"/>
4   <e n1:a="x:a" xmlns:n1="http://w3.org/2"/>
5   <n1:e a="a" n2:a="x:a" xmlns:n1="http://w2.org/2"
6     xmlns:n2="http://w3.org/2"/>
7   <e n1:a="y:a" n2:a="x:a" xmlns:n1="http://w3.org/1"
8     xmlns:n2="http://w3.org/2"/>
9   <e n1:a="x:a" n1:b="x:b" xmlns:n1="http://w3.org/2">
10    <n1:e n1:a="x:a" n1:b="x:b" xmlns:n1="http://w3.org/2">
11      <n3:e n1:c="n1:c" n2:a="x:a" n2:b="x:b"
          xmlns:n1="http://w2.org/1"
          xmlns:n2="http://w3.org/2"
          xmlns:n3="http://w3.org/5"/>
12    </n1:e>
13  </n1:e>
14 </n1:doc>
```

XML document after transformation:

```
1 <n1:doc xmlns:n1="http://w3.org/2">
2   <n1:e a="a"></n1:e>
3   <n1:e n1:a="x:a"></n1:e>
4   <e n1:a="x:a"></e>
5   <n1:e xmlns:n1="http://w2.org/2" xmlns:n2="http://w3.org/2" a="a"
6     n2:a="x:a"></n1:e>
7   <e xmlns:n1="http://w3.org/1" xmlns:n2="http://w3.org/2"
8     n1:a="y:a" n2:a="x:a"></e>
```

```

7         <e n1:a="x:a" n1:b="x:b"></e>
8         <n1:e n1:a="x:a" n1:b="x:b">
9             <n3:e xmlns:n1="http://w2.org/1" xmlns:n2="http://w3.org/2"
xmlns:n3="http://w3.org/5" n1:c="n1:c" n2:a="x:a" n2:b="x:b"></n3:e>
10         </n1:e>
11     </n1:doc>

```

After transformation in an XML document:

- ***empty spaces inside open tags (lines 5, 6, 9) have been normalised;***
- ***excessive namespace declarations (lines 2, 3, 4, 7, 8) have been removed;***
- ***empty elements have been transformed into pairs of open/closed tags (lines 2–7, 9);***
- ***lexicographical order for namespace and attribute declarations (lines 5, 6, 9) has been established inside components.***

–

10. Description of Base64 encoding algorithm

This annex describes the encoding algorithm [base64]. This algorithm applies to the values of ACs or SCs and also to an EM (EM packet) when generating/verifying ACs.

Identifier of the algorithm describing the transformations:

“<http://www.ietf.org/rfc/rfc2045#base64>”.

N o t e : The annex contains an overview of the algorithm. Special features of the algorithm are presented in the original specification [base64].

The algorithm [base64] was developed for presentation of octet sequence in a form that is not intended to be read by a human. The encoding and decoding algorithms are simple; however, the encoded data occupies a larger volume compared to the initial data by approximately 33%.

For encoding, 65 US-ASCII symbols are used, allowing to present a 6-bit sequence as a printed symbol (the 65th symbol is used as a filler).

The encoding algorithm represents a 24-bit group of input bits in the form of an output string of 4 symbols. Input flow processing is performed from left to right, and a 24-bit group is generated by concatenation of three 8-bit input groups. The 24 bits thus obtained are treated as four combined 6-bit groups, each of which is translated into a separate symbol of the Base64 alphabet.

Each 6-bit group is used as an index for the array of 64 printed characters. The symbol obtained according to the index is placed in an output string. Таблица F.1 demonstrates symbols used in encoding that make up the Base64 alphabet.

T a b l e 5. Base64 Alphabet

Index	Symbol	Index	Symbol	Index	Symbol	Index	Symbol
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	Filler	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

Figure 1 demonstrates the operation of the encoding algorithm [base64].

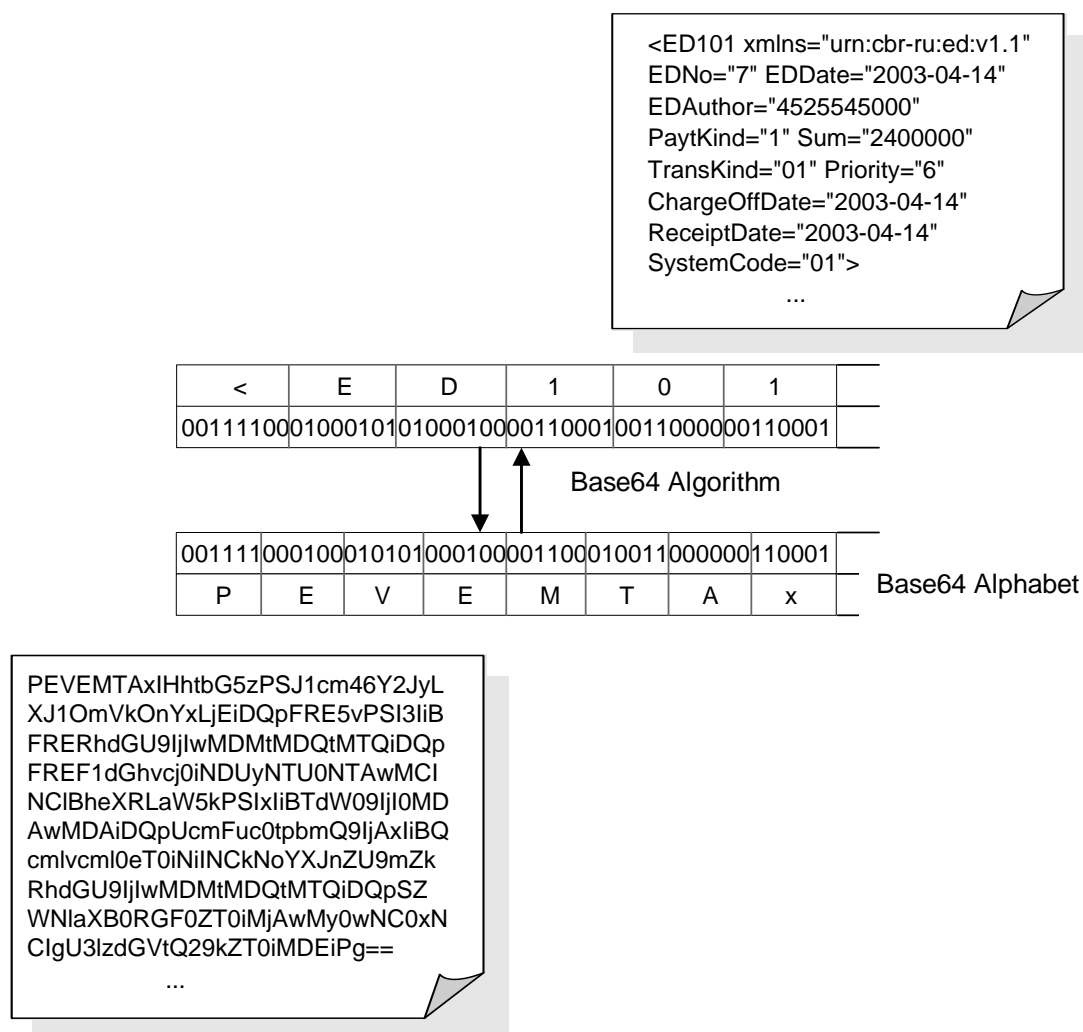


Fig. 1. Demonstration of the operation of the Base64 encoding algorithm

Output flow may be presented by strings not exceeding 76 symbols in each. All symbols of the string end and other symbols not represented in the Base64 alphabet shall be ignored when decoding.

If the last group of bits in an input flow contains less than 24 bits, special processing is required: zero bits are added to the right until a whole number of 6-bit groups is generated. If the result fewer than four 6-bit groups are generated, the missing groups are replaced with filler symbols (=). As an input flow always contains a whole number of octets (8-bit groups), there are three options:

- The final group in an input flow contains 24 bits precisely. In that case, the final group in the output flow contains 4 symbols without filler symbols.
- The final group in an input flow contains 8 bits precisely. In that case, the final group in an output flow contains 2 symbols followed by two filler symbols (=).
- The final group in an input flow contains 16 bits precisely. In that case, the final group in an output flow contains 3 symbols followed by one filler symbols (=).

As the symbol "=" is used only as a filler, the occurrence of this symbol can be a sign that the end of the data has been reached. It should be remembered that if the number of input octets is a multiple of three, the symbol "=" in an output flow will never occur.

Any symbols outside the Base64 alphabet shall be ignored in data encoded pursuant to the algorithm [base64].