Technical Documentation

Archetypes in V2

Medical-Objects has been involved in developing a standard way to encode CEN-13606 or OpenEHR archetypes in HL7 V2. This significantly enhances the ability on HL7 V2 to participate in an electronic health record system. Some documentation of this is available here

Technical Documentation

One of the advantages of HL7 is to allow interoperability between systems from different vendors, running on different operating systems. To allow this the interfaces need to be documented and constrained. Documents on this page contain technical documentation of Medical-Objects interfaces and are only likely to be of use to people with a good understanding of HL7.

Digital Signature Documentation

Problem: Signatures and ORU/Order messages

To Quote from the Electronic Transactions Act 1999

1.1 Where a medical practitioner:

Refers a patient to a specialist or consultant physician: or

Requests a pathology service; or

Requests a diagnostic imaging service,

by electronic means, HIC requires that the referral or requests:

a) Consist of either a standard e-mail message* , or a recognised EDI format,

that is transmitted using HIC's PKI standards;

b) Meet all the requirements for a Referral or Request provided in the Act, the

regulations and the HI(PS) regulations; and

c) Be digitally signed by the referring/requesting practitioner using an individual

private key for which there is a current public key certificate recognised by the

HIC (in accordance with HIC's PKI standards), to allow a specialist,

consultant physician, or Approved Pathology Practitioner or medical

practitioner to verify the authenticity of the Referral or Request upon receipt.

While individual messages can be signed, it makes batch transmission and passage through interface engines very difficult unless the message is sent as a single message and the whole message is archived. This also means that being able to verify the authenticity of a message requires this file to be made available.

It would be much better to embed the signature in the message so that only the data in the message was required to verify the authenticity at any stage in the future. By using standard structures already in the standard this data could reside in a message and be ignored if verification was unavailable or not desired. Old applications would simply process the data in the standard way without being able to interpret it but without the need for any changes. The data remains available for later use then.

From the point of view of the doctor and the HIC the important data is content of the message, who it was on and what date it was created. Data, such as name data is volatile over time and should be encapsulated within the message so that the document can be sent under eg a married name without invalidating the signature.

This is a scheme to do that. It is a tested and proven schema. It covers PKI signatures and MD5 and SHA1 hashes. Hashes do not meet the ETA requirements but may be useful when you wish to protect the integrity of the data for other reasons.

The basic algorithm is to add 2 OBX segments to the end of an ORU message. The first added OBX if FT (freetext) and contains the important PID and OBR data such as the patient name, the test name and the date of the test. This can be extended at will as it has to be generated prior to signing the message and its actual content is not critical, basically whatever data you wish to preserve over time. The second OBX contains the actual signature data, which in the case of the HIC PKI is encapsulated in a ED segment as it is binary. what is actually signed is all the OBX data above the Last segment. This will include the first OBX that was added.

eg: A simple Message:

MSH|^~\&|TEST^TESTAPP^L|Buderim GE Centre^7C3E3681-91F6-11D2-8F2C-444553540000^GUID|||20040410141133||ORU^R01...

PID|1||||PATIENT^Test^^^^^^L||20000101||||139 King Street^^BUDERIM^QLD^4556^AU^C|AU

PV1|1|O|||||0341615J^WHITE^MELISSA^^^DR^^^AUSHICPR^L^^^UPIN|0341615J^WHITE^MELISSA^^^DR^^^AUSHICPR^L^^^UPIN|||||||N

ORC|CE||F71DEE61-D19E-4571-AF7B-BF8C74597CAB^Buderim GE Centre^7C3E3681-91F6-11D2-8F2C-444553540000^GUID||CM|||||||0341615J^WHITE^MELISSA^^^DR^^^AUSHICPR^L^^^UPIN

OBR|1||F71DEE61-D19E-4571-AF7B-BF8C74597CAB^Buderim GE Centre^7C3...

OBX|1|FT|28655-9^^LN||This a simple \H\Test Message\N\ To demonstrate signing a...

OBX|2|SN|5048-4^ANA titre^LN||<^40|titre|||||F

now it is signed by a PKI key:

MSH|^~\&|TEST^TESTAPP^L|Buderim GE Centre^7C3E3681-91F6-11D2-8F2C-444553540000^GUID|||20040410141133||ORU^R01|

PID|1||||PATIENT^Test^^^^^^L||20000101||||139 King Street^^BUDERIM^QLD^4556^AU^C|AU

PV1|1|O|||||0341615J^WHITE^MELISSA^^^DR^^^AUSHICPR^L^^^UPIN|0341615J^WHITE^MELISSA^^^DR^^^AUSHICPR^L^^^UPIN|||||||N

ORC|CE||F71DEE61-D19E-4571-AF7B-BF8C74597CAB^Buderim GE Centre^7C3E3681-91F6-11D2-8F2C-4445535400.....

OBR|1||F71DEE61-D19E-4571-AF7B-BF8C74597CAB^Buderim GE Centre^7C3E3681-91F6-11D2-8F2....

OBX|1|FT|28655-9^^LN||This a simple \H\Test Message\N\ To de....

OBX|2|SN|5048-4^ANA titre^LN||<^40|titre|||||F

OBX|3|FT|SIGNATURE_HEADER^^L||PKI Signed Message\.br\Patient: PATIENT, Test DOB:01.01.2000 \.br\Report: Physician Discharge Summary Dated: 10.4.2004\.br\Signed: 10/04/2004 2:28:57 PM||||||F

OBX|4|ED|AUSETAV1^PKI Signature^L||AUSHICPKI^AP^Octet-stream^Base64^MIIKTgYJKoZIhvcNAQcCoIIKPzCCCjsCAQExCzA.....|||||F

OBX 3 and 4 have been added. OBX 3 is simply to store relevant ORC/OBR/PID data that needs to be signed. An application that doesn't understand the signature should still cope with these segments.

To look at what is signed - which is where the algorithm comes in: (see file 'SDataETA.txt')

FT.28655-9..LN.....F..This a simple \H\Test Message\N\ To demonstrate signing and \H\ORU\N\ message\.br\\.br\Another Line \.br\\.br\A few encoded characters \F\\S\\T\/\.br\\.br\The end.

SN.5048-4.ANA titre.LN..titre....F...<.40...

FT.SIGNATURE_HEADER..L.....F..PKI Signed Message\.br\Patient: PATIENT, Test DOB:01.01.2000\.br\Report: Physician Discharge Summary Dated: 10.4.2004\.br\Signed: 10/04/2004 3:05:10 PM.

What is extracted are all the fields in OBX that affect display. The reason for doing it this way is to overcome problems with and errors in formatting (eg unneeded Field separators) that may be in the original message and to allow any system that stores the data to reconstruct this at will so the signature can be validated at any time. There is no requirement that the data stay in HL7 format, just that the data is preserved and available. I have tested it on XML and classic encoding and you can translate freely and still have a valid signature.

The exact same algorithm is used for MD5 hash and SHA1 hash versions. You can run a separate MD5 hash checker over the 'SDataMD5.txt' file and get the same hash value as in the message (eg http://www.whitsoftdev.com/md5/)

A similar process could be used for orders. What was ordered and who it was ordered by can be added to the SIGNATURE_HEADER and an OBX added to the order message that signs this data. Then you can "verify the authenticity of the Referral or Request" at any time. The signature data can be stored in the same way existing data is stored.

The required Identifiers are:

AUSHICPKI

To indicate that this is a HIC PKI signature - used in ED segment – Namespace ID

AUSETAV1

To indicate that this OBX contains PKI signature data - should have a version number to indicate algorithm (ED Segment)

AUSSHA1HASH

To indicate that this is a SHA1 hash of the message (Base 64 encoded in ST segment as short)

AUSMD5HASH

To indicate that this is a MD5 hash of the message (in ST segment as usually Hex encoded)

The required Identifiers are:

Example files are included that show the 3 variations. The original message is in TestPatient.oru

FileName

Signature Algorithm

What was signed

TestpatientETA.oru

PKI Signature (ED Datatype)

SDataETA.txt

TestpatientMD5.oru

MD5 hash (ST Datatype)

SdataMD5.txt

TestPatientSHA1.oru

SHA1 hash (ST Datatype)

SDataSHA1.txt

The Algorithm

The algorithm treats each OBX in a standard way, obviously each datatype has to be treated differently but basically every field is separated by a '.' (Full stop) and each OBX segment by a <CR><LF> sequence. If a field is blank then the fullstop is still written. A few fields of some datatypes have been omitted, usually because of optional, non critical values in these fields. In general ST/FT and SN will cover 99% of the segments but it is possible to encode any datatype. The order of the fields in the specification is used to write the values, with the one exception of the Observation value in OBX which is written last for every OBX. Pseudo code for this is detailed below.

Build a string to sign or hash

For each OBX in order add these values (include SIGNATURE_HEADER OBX)

{

OBX.Value_Type

'.'

OBX.Observation_Identifier.Identifier

'.'

OBX.Observation_Identifier.Text

'.'

OBX.Observation_Identifier.NameOfCodingSystem

'.'

OBX.Observation_SubID

'.'

OBX.Units.Identifier

'.'

OBX.Units.Text

'.'

OBX.Units.NameOfCodingSystem

'.'

OBX.References_Range

'.'

for each OBX.Abnormal Flags

OBX.Abnormal_Flags[i]

'.'

OBX.Status <- or 'F' if blank

'.'

OBX.Date_Of_Observation

'.'

Then add all field values for Result values

Depends on value type but for example

SN - Structured Numeric

SN.Comparator

'.'

SN.Num1

'.'

SN.Separator

'.'

SN.Num2

'.'

Now add <CR><LF>

}

Now check or create detached signature/hash on resulting string

OBX Value Type Handling

SN as above

FT/ST/DT/TS

for each observation value

Add Text

Add '.'

These are less common but could appear

XCN

for each observation value

Add IDNumber

Add '.'

Add FamilyName

Add '.'

Add GivenName

Add '.'

Add MiddleName

Add '.'

Add Suffix

Add '.'

Add Prefix

Add '.'

XPN

for each observation value

Add FamilyName

Add '.'

Add GivenName

Add '.'

Add MiddleName

Add '.'

Add Suffix

Add '.'

Add Prefix

Add '.'

ED

for each observation value

Add SourceApplication.NameSpaceID

Add '.'

Add SourceApplication.UniversalID

Add '.'

Add SourceApplication.UniversalIDType

Add '.'

Add MainTypeData

Add '.'

Add DataSubType

Add '.'

Add Encoding

Add '.'

Add Data

Add '.'

RP

for each Observation value

Add Pointer

Add '.'

Add TypeOfData

Add '.'

Add ApplicationID.NameSpaceID

Add '.'

Add ApplicationID.UniversalID

Add '.'

Add ApplicationID.UniversalIDType

Add '.'

Add SubType

Add '.'

EI

for each Observation value Add EntityIdentifier

'.'

Add NameSpaceID

'.'

Add UniversalID

'.'

Add UniversalIDType

'.'