Macedonian Interoperability Building Block: **IOP-T**

(Technical Interoperability)

ANNEX: MIM Protocol

Johann Höchtl, Bernd Zwattendorfer, Peter Reichstädter, Igor Crvenov, Filip Manevski, Nadica Josifovski

Delivery: 13.08.2015
Version: V1.0

This project is funded by the European Union

TABLE OF CONTENTS

1 Version History 3
2 Abstract 4
3 Definition of Terms 5
4 Problem Statement 6
5 Target Audience 7
6 Features 8
7 Responsibilities 9
8 MIM Architecture components 10
8.1 Pre-requisites for MIM 10
8.2 Components of MIM 10
8.3 MIM Parameters 12
8.4 MIM Metaservices 14
8.5 Error handling 16
8.6 MIM Metaservice SOAP container 18
9 MIM Messaging and Configuration 28
9.1 Principles and Standards 28
9.2 Service access control 28
9.3 Providing a service 29
9.4 Service discovery 29
9.5 Message Flow 33
Informative: The CC as service consumer and service provider 36

This project is funded by the European Union

1 Version History

Version	Remarks
2015-05-15, 1.0	**Final**
2015-07-15, 1.0	Changes according to input of IOP system implementations Clarification on authentication and authorization **Final**
2015-07-27, 1.0	Changes according to input of IOP system implementations and common discussion on 2015-07-24 **Final**
2015-08-12, 1.0	Changes according to clarifications MK-TK and KING-ICT: XSD Schema, SOAP envelope examples **Final**

This project is funded by the European Union


2 Abstract

This document specifies the Macedonian Interoperability Protocol MIM.

MIM is

- a specification of a message protocol with minimum domain specific semantics;
- a set of services to be implemented by an interoperability system, enabling the working together of information systems in an interoperable manner;
- the description of a message flow between defined services using a specified protocol.

While the semantics of MIM is implementation independent to ease readability and applicability this specification assumes the MIM services to be implemented as web services and message exchange using SOAP.


This project is funded by the European Union

3 Definition of Terms

Abbreviation	Meaning
MIM	Macedonian Interoperability Framework, Macedonian Informatics Magistrale
OPS	Organisation Providing Service
MSC	Metadata Service Catalogue

This project is funded by the European Union


4 Problem Statement

For information systems to exchange information in a meaningful way it is necessary to define a minimum set of agreed components, formats and procedures:

1. Message formats;
2. metaservices, which perform unambiguous operations on these messages; and
3. the interplay of required infrastructure services.

Message format, metaservices and infrastructure components, which enable the administration of actual services provided by parties participating in an interoperability system, must fulfil these common requirements:

1. They have to be well defined;
2. They have to be technology agnostic;
3. They have to be reliable;
4. They have to be secure;
5. They have to use well defined standards for which lots of available knowledge exists.

Services must result in building blocks which enable an interoperability eco-system which is resilient, fault tolerant, easy to implement and easy to administer.

The resulting system architecture must clearly separate and differentiate between application logic and communication logic and empower interoperability participating parties to continue considering services within their premises as “their world”, retaining sovereignty.


This project is funded by the European Union


5 Target Audience

This document is targeted towards parties willing to participate in the MIM and which are concerned with detailed technical aspects of the interoperability system. They will typically be in these positions:

- CIOs which are confronted by ever increasing requirements which have to be fulfilled with diminishing budgets and which understand that collaboration in the mid to long term is cheaper and more future proof than to continue working in silos.
- System integrators: They will have to check the required components which enable the MIM architecture, identify already existing ones, and define the required efforts to adapt these existing components or to acquire new ones;
- System developers: They will have to adapt or implement interfaces towards generic service endpoints defined by MIM;
- Security Officers: They will have to check the specification for security soundness and if the resulting system fulfils defined security requirements.


This project is funded by the European Union

6 Features

MIM provides certain feature levels. Feature level 1 includes synchronous communication. Feature level 2 includes synchronous and asynchronous communication.
Every IOP-system must at least implement feature level 1.

Feature	Description	Feature level
Built in Security	Data: Based on X.509 V3 certificates. Signature and Encryption of messages Message: Verification of the eligibility of a caller to call a service of a provider	1
Routing	Of messages to all entities which are known in the organisation providing service	1
Logging	Messages: Fact at what time entity A sent a message to entity B Errors:	1
Message Storage	In feature level 1, no messages will be stored on the IOP system (including CC and CS). Feature level 2 will store messages to be retrieved by a caller at a later time (asynchronous communication) or store messages which will be delivered in case of failing IOP systems (CC to CS or CS to CC).	1
Error handling	Error handling is provided by MIM on the communication layer and reports technical errors concerning the communication of messages on the IOP system such service unavailability or attempts of service access violations.	1
Different Communication patterns	Synchronous and asynchronous	2

This project is funded by the European Union


7 Responsibilities

While MIM is a technical specification for interoperability in the Macedonian administration, some organisational responsibilities have to be met in order to establish trust among the participating parties.

- MISA is responsible for the administration and provision of the organization providing service OPS.
- An Interoperability Bus Provider is responsible to implement a communication server CS according to the MIM specification.
- MISA is responsible to certificate Interoperability Bus Providers which implemented an Interoperability Information System.


This project is funded by the European Union

8 MIM Architecture components

8.1 Pre-requisites for MIM

For the MIM Architecture to work, these components have to be in place:

- Organization Provider Service OPS: This service provides a list of organizations. As a minimum, the organizations clear name and a unique ID has to be provided. This information typically comes from a file or a central service(e.g., LDAP-service). The details of how this information is provided are outside the scope of the MIM architecture.

8.2 Components of MIM

The MIM architecture consists of these components:

Acronym	Description	Description

CS	Communication server	A communication server is responsible for:
		• Routing: messages from service clients to service providers and vice-versa;
		• Logging: the fact of message exchange and who/which organisation participated in the message exchange
		• Security: Only messages according to the MIM specification are transferred.
		• Message storage:
		  ○ In MIM level 1, no messages will be stored on CS except for the purpose of immediate message delivery.
		  ○ In MIM level 2, messages will be kept either on the CC or the CS for the purpose of asynchronous communication or if a service participating in the IOP system is not available to receive a message in a timely manner.
		
		Every CS must implement a CC endpoint, which must only be accessed by another CS.
		
		For an entity to participate in the MIM it is required to contract with an authorized MIM service provider.
		
		In order to become an authorized MIM service provider, certification from MISA is required.

CC	Communication client	A communication client is responsible for: • Interoperability: All messages from a MIM participating party must go through the CC. • Metaservices: The CC exposes a set of

This project is funded by the European Union

			metaservices which enable the administration of services an entity is willing either to consume or to provide in the MIM Architecture. • Security: The CC signs the MIM message and encrypts the message body. As all messages exchanged on the MIM system have to go through the CC, the CC acts both as a client as well as the provider of services. A party willing to participate in the MIM system can either implement a communication client on its own or obtain one from an authorized and certificated MIM service provider.
CCext	'External' client	Communication	There is only one external CC per service bus. These are virtual communication clients (or applications) on the (central) Communication server (CS) and are interfaces to the other (or any other for that matter) service bus. The two service busses will communicate to each other only through these external CCs.
BS	Backend Service		A service which participates in interoperability using MIM, either as a service consumer (client) or as a service provider. • The service will have to communicate with the CC. • The service must not communicate with the CS. • If the BS is to communicate with other backend systems it is in the responsibility of the BS provider by technical and organizational means that only authorized persons can use the BS and that all depended systems at least fulfil security requirements imposed by MIM and legal obligations (principle of least security[1]).

A MIM system provider may provide additional components beside the mentioned ones, however these components are implementation dependent and outside the scope of the MIM specification[2].

[1]The maximum security level of individual systems participating in an interoperability system must be the minimum security level of the resulting, virtual system.

[2] For example a service configuration and service revocation web application, logging and auditing service, etc.

This project is funded by the European Union


8.3 MIM Parameters

The MIM Protocol and for the MIM Metaservices to function, a set of parameters is required. These parameters are:

Parameter	Description
Consumer	Identifier of a service consumer.
Provider	Identifier of a service provider. The distinction between provider and consumer is valid within the context of a communication and makes sense when thinking of a communication direction. If no communication is happening with the MIM system, provider and consumer are equal to a MIM participating party.
RoutingToken	Is the identifier of the provider; In case the provider is located on other ESB it is precededby the identifierof the ESBfollowed by $$ (=2 "dollar signs").
ServiceId	Identifier of a service (of adequate provider).ServiceId is a unique identifier of the service, not the whole URL including methods. The SeviceId might unambiguously identify the Business service provided by Provider for those cases, where ServiceMethod is empty or not present.
ServiceMethod	Identifier of a Method provided by ServiceId. This field is optional and used in cases where the client might hold all the required information to address a final Business service providing endpoint.
TransactionId	Unique identifier for a transaction. The TranscationId is a GUIDv4 generated by consumer'sCC.
CorrelationId	Implementation specific identifier issued by a BS to correlate messages in the back end system.
Dir	Represents direction of message flow, either "Request" or "Response".
callType	Method call type. Values are "synchronous" or "asynchronous". MIM level 1 only implements synchronous calls on the communication layer whereas MIM level 2 supports asynchronous calls to services which by a BS are only provided in an asynchronous manner.
publicKey	public key of an entity which issued a service call in base64encoding originating x.509. This public key might either be issued to an institution, an organizational unit or a person.
Status	The status of the communication. Standard HTTP status codes[3] are used with meaning described further below.
StatusMessage	List of Messages of the IOP System. The interpretation of the message might be dependent on the value of the Status field. The encoding and serialization of StatusMessage is IOP system dependent.

[3] http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html


This project is funded by the European Union

Message	The message payload, always encrypted.
Signature	The signature value of the MIM message
MimeType	Mime Type of the message payload. For example application/soap+xml or application/OCD[4]. A complete list of defined mime types is available from IANA[5].
TimeStamp	A timestamp issued by the Interoperability System. Signed timestamps issued by a TSA are preferred.
Extension	Placeholder for future use.

Status IDs are standard HTTP error codes with more details (such as expired PKI certificates) provided in the error message in the body and optionally in the MIM header field StatusMessage.

MIM defines the following Status codes with the following meanings:

Status	Meaning	MIM level
102	Transaction in progress. The request cannot be fulfilled as the system is still processing a request (used to signal a caller that an asynchronous call is still in progress).	2
200	Message transport successful	1
400	The invocation of one of the MIM provided service methods contained an invalid combination of parameters. The body will contain additional information.	1
401	The invocation of a business service exposed by a CC could not be fulfilled, because the issuing CC couldn't be authorized.	1
402	The invocation of a business service exposed by a CC could not be fulfilled, because the receiving CC determined that business requirements to fulfil the request have not been met. This might include, but is not limited to, pending service usage payments	1
403	The invocation of a business service exposed by a CC could not be fulfilled, because the IOP system could not find a valid route to the service providing CC.	1
429	This Status code **might** be sent from the CC if fraudulent actions like DoS attacks are recognized.	1
500	The IOP system experienced an internal server error	1
503	Parts of the IOP system are not available such a CC not being able to reach the CS	1

[4]Specification of the OmnifariousContainerforeDocuments (OCD) http://www.gov2u.org/projects/spocs_starter_kit/images/files/D2.2_Standard_document_and_validation_common_specifications.pdf

[5]http://www.iana.org/assignments/media-types/media-types.xhtml

The list of status codes might be extended at a later point in time by the organizational means as laid out in the IOP-O.

8.4 MIM Metaservices

Metaservices are part of MIM and enable the enlisting and revocation of services, message retrieval and error handling. The following list of services enables entities to participate in the MIM[6]:

Service name	Description including input Parameters and return values	MIM Level
RegisterService	Used to register a service to the MIM. Input: • ServiceId: Unique identifier of service in provider environment. • WSDL – XML document in WSDL standard that contains definitions for methods behind provided ServiceId. EndPoint addresses of services stated in WSDL are pointing to local network. Return: None	1
UnRegisterService	Used to unregister a service in MIM Input: • ServiceId - Unique identifier of service. Return: None	1
GetProviders	List of providers for metaservice call invoking consumer that are exposing at least one service for that consumer. Input: None Return : • List of service provider s containing: ○ identifiers as RoutingToken; ○ Public key of the service providing organization to be used end-to-endpoint message encryption facilitated by the CC.	1
GetServices	Returns for a specific provider the list of provided services.	1

[6]In cases where the return message is None, the header of the message invocation will still be set according to the specification and the body might contain data in cases where an error will have to be returned.


This project is funded by the European Union

	Input: • ProviderId – Unique identifier of provider. This is the ID of the organization to which service access has been granted by other organization and which correlates to the ID used for uniquely identifying organisations in the MIM system. Return: • List of service identifiers uniquely identifying a service. In those cases where Services are returned from another MIM system, the service designator contains routing information to uniquely identify services hosted on another MIM system (RoutingToken assembly and interpretation defined further below).	
GetService	Returns service definitions (WSDL document) for a specific service. Input: • ProviderId – Unique identifier of provider. • ServiceId– Unique identifier of provider's service. • callType: This parameter influences how a call to the service, once it is invoked on the CC, will react.Implementations of MIM level 1 are save to ignore this parameter, and will always respond with standard WSDL of the requested business service. Return: • The returned value is a XML document according to WSDL standard.	1,2
ListConsumers	Returns list of all registered consumers. Input: • ServiceId– Unique identifier of provider's service. Return: • List of consumer identifiers.	1
CheckStateByTransactionId	Get state of transaction, useful in asynchronous call scenario Input: • TransactionId– Unique identifier of transaction.	2


This project is funded by the European Union

	Return:
	• Numerical ID specifying state of request; Possible values are:
	  ○ Pending
	  ○ Unknown
	  ○ Deliverable	

GetMessageByTransactionId	Gets message by TransactionId from CC that is called in asynchronous mode. A call to this method will delete the message from the message store. This method is called by the backend system BS to retrieve messages returned by asynchronous service invocation. Input: • TransactionId – identification of a transaction Return: • Response message	2
PostMessage	Posts a message into the MIM system. This method may be called to post a message into the MIM system which will be stored in the message queue and can be retrieved by GetMessageByTransactionId in those cases, where the providing service doesn't return in time to the invocation of the CC. Input: • TransactionId of asynchronous service call invocation for which the response shall be posted; • Direction as parameter "dir": Direction of the message, either "Request" or "Response" • Message to be posted After PostMessage has been called, a call of CheckStateByTransactionId with a matching TransactionId will return state "Deliverable" and the message can be retrieved by GetMessageByTransactionId	2

While the MIM metaservices are agnostic to technology and in theory could be provided using JSON-API, RPC or Websockets, a common usage scenario is to use web services. Thus the aforementioned MIM parameters will be further detailed in the context of SOAP.

8.5 Error handling

Communication error handling is an integral feature of MIM. If a request cannot be successfully processed, the aforementioned Metaservices will return a status code in the field Status of the message header indicating the unsuccessful communication attempt or e.g. internal server error. In

case of internal errors of the interoperability system, standard SOAP errors should be transmitted. Optionally, to ease a centralized view on the status of the system, the field StatusMessage can contain further information, which might be interpreted specific to the implementation of the IOP system.

The Soap fault message body will contain detailed error information, containing at least:

- DateTime: datetime stamp according to ISO 8601
- Location: error location to be uniquely interpreted within the domain where the error occurred
- Message: error message;
- MessageID: Implementation specific messageID[7].

The structure of SOAP 1.2 fault message is:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:m="http://www.example.org/timeouts"
              xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>m:MessageTimeout</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">Sender Timeout</env:Text>
      </env:Reason>
      <env:Detail>
        <m:MaxTime>P5M</m:MaxTime>
      </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

The encoding of the error message will be defined by the MimeType.

[7]Note that a system wide specified Status is set in the header and will be used to separate successful method invocations from erroneous returns.


This project is funded by the European Union

8.6 MIM Metaservice SOAP container

In a SOAP context, the namespace of the MIM protocol shall be set to http://mioa.gov.mk/interop/mim/v1.

SOAP envelope consists of a SOAP header containing a MIM Header "H", a MIMadditionalHeader "A" and a CryptoHeader "C" and a SOAP body containing a MIM Message Body "B".

The SOAP message has to follow this example:

```
<?xmlversion="1.0"?>
<soap:Envelopexmlns:soap="http://www.w3.org/2003/05/soap-
envelope"xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:mioa="http://mioa.gov.mk/interop/mim/v1">
<soap:Header>
<mioa:MIMHeader/>
<mioa:MIMadditionalHeader/>
<mioa:CryptoHeader/>
</soap:Header>
<soap:Body>
<mioa:MIMbody>
            <Message/>
      <mioa:MIMbody>
</soap:Body>
</soap:Envelope>
```

A MIM Body may contain an encrypted message body or a plain text message body, but not both at the same time.
If a message is encrypted, the CryptoHeader-Element must be used.

MIMHeader/MIMBody parameter description

An occurrence of M means the field has to be mandatorily provided, depending on the context of the call (request vs. response). O denotes optional occurrence.

Parameter	Header H / Body B	Occurrence	xsd Data Type	Usage / Origin
Consumer	H	M	string	Set by the CC the moment a message invocation is received, identifying the message invocation originator
Provider	H	M	string	Set by the CC the moment a message response is sent, identifying the origin of a service providing organisation(Provide r is filled only on

This project is funded by the European Union

					response (CC on provider's side)).
RoutingToken	H	M		string	The RoutingToken is obtained by the method GetProviders. A detailed description of the meaning of the RoutingToken is given in 9.2.1
Service	H	M		string	The name of the service is injected by the CC when a request for business service invocation is received and does not include the method(s) name. The Service might also be unambiguously identified by an ID (ServiceId) which is only resolvable by the providing CS.
ServiceMethod	H	O		string	The name of the method provided by Service. This field is optional. If present, the combination of Service and Method defines the Business functionality provided by Provider unambiguously.
TransactionId	H	M	string, containing a UUIDv4, validating against xsd:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}"		The transactionId is injected by the CC into the MIM header the moment a service invocation is received.
Dir	H	M		string	The direction of a message. The direction identifier will be derived from the context of a message whether it is a service request or a response. If the


This project is funded by the European Union

				communication client constructs a SOAP/MIM request, the direction is set to “Request”. If the communication client constructs a response message, the direction is set to “Response”. In the case of asynchronous communication, a call to GetMessageByTransactionId will set the direction to “Response”. A call to PostMessage will set the direction depending • On the received value of Dir in the MIM header; • on the query parameter “Dir”.
callType	H	M	enumeration, containing “asynchronous”, “synchronous”	MIM level 1 will only support synchronous communication on the communication layer.
publicKey	H	O	string	The public key of an entity invoking a service must be injected by the CC
Status	A	M	string	The status of the transportation should be obtained from the underlying transportation technology. In case of https-Transport this is the http status code.

This project is funded by the European Union

Message	B	O	xml	The payload body of the response. It is the CC's response to perform signing and encryption of the body.
MimeType	H	O	string according to http://www.iana.org/assignments/media-types/media-types.xhtml	The mime type of the Message. Setting the mime type is implementation specific and can be obtained in several ways: 1. Copying it verbatim from the custom MIM header field MimeType once a response message is to be transmitted over a CC; 2. Using a heuristic which will determine the mime type from the message body[8]. 3. Setting it from the URL parameter "MimeType" passed to the business service configured on the CC endpoint 4. Obtaining it from the environment

[8] http://stackoverflow.com/questions/58510/using-net-how-can-you-find-the-mime-type-of-a-file-based-on-the-file-signature

This project is funded by the European Union

TimeStamp	H	M	xsd:dateTime	The timestamp TS in the MIM header field is set on the CC. It is advised to set the TS upon logical completion of the MIM message, i.e. before signing and encryption. The Timestamp must not be set to the time of actual delivery but to the time of first delivery attempt[9].
StatusMessage	A	O	xsd:anyType	See above. The StatusMessage must not be part of the message signature.
CorrelationId	H	O	xsd:string	A backend-specific identifier to correlate the state of the backend-issuing invocation to the call of the IOP-system. The value will be set on the CC 1. By setting the custom header field in the BS when invoking the CC exposed business service. 2. from the URL parameter "CorrelationId" 3. Obtaining it from the environment
Signature	H	M	xsd:string	Contains the signature of the MIM message

[9] If a service is caching a message because a receiving system is not responding to a request in a timely manner, the TS is set to the time the MIM message has been logically assembled and saved into the message store for later delivery, not to the date of actual delivery. Actual time of delivery shall be recorded in the logging facility of the IOP system.

This project is funded by the European Union

These MIM headers fields must be injected into the SOAP headerby the CC receiving a request to be forwarded to the CS:

- Consumer – Identifier of the institution as obtained from the OPS
- Service – ServiceId as registered in the MSC
- publicKey – optional
- TransactionId
- Dir – for MIM level 1 this will always be “Request”
- Timestamp
- Signature of requesting organisation
- Message

The following MIM headers might already be set in the custom MIM header upon reception on CC or injected into the SOAP specific MIM headers by copying from web service query parameters as they are received on the WS endpoints exposing MIM controlled business services:

- CorrelationId

The following MIM header parameters must be injected by the CC upon the completion of a service request:

- Provider (Provider is filled only on response (CC on provider’s side))
- Status
- MimeType
- Dir – for MIM level 1 this will always be “Response”
- TimeStamp
- Message
- Signature of the responding organisation

Upon message response, these header fields will be copied verbatim from the original message request:

- Consumer – Identifier of the calling institution as obtained from the OPS
- Service – ServiceId as registered in the MSC
- publicKey – optional
- TransactionId

Upon final assembly of the MIM message, the message signature must be calculated and put into the Signature header field.

Header parameters must be part of the signed message except in the case of the optional parameter StatusMessage which might be dynamically appended outside MIM header(MIMadditionalHeader) (not part of the signature) by the IOP infrastructure as the message passes through the system (compare also with example provided above).


This project is funded by the European Union

CryptoHeader element

Currently for the needs of encryption and decryption of the request/response body of each message symmetric encryption is used. In order to exchange keys for symmetric encryption between consumer and provider and vice versa CryptoHeader element is introduced with 2 elements:
Key: Encrypted Symetric Key further used in symmetric encryption of the request/response body. This value is randomly generated for each request and then encrypted using the Public Key of the recipient of the message with RSA algorithm. In this way only the owner of the matching private key is able to decrypt this value and use it for decryption of the body.
Initialization Vector:
For a given secret key k, a simple block cipher that does not use an initialization vector will encrypt the same input block of plain text into the same output block of cipher text. If you have duplicate blocks within your plain text stream, you will have duplicate blocks within your cipher text stream. If unauthorized users know anything about the structure of a block of your plain text, they can use that information to decipher the known cipher text block and possibly recover your key. To combat this problem, information from the previous block is mixed into the process of encrypting the next block. Thus, the output of two identical plain text blocks is different. Because this technique uses the previous block to encrypt the next block, an initialization vector is needed to encrypt the first block of data.
The request/response body is encrypted with The Advanced Encryption Standard (AES), also referenced as Rijndael.
https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
Intorduction of this mechanism ensures that there is no overhead both in performance and size of the encrypted result.

Find an example for the CryptoHeader element in the (following) XSD Schema.

MIMadditionalHeader Element

The following parameters: Status, Status Message, ProviderEndpointUrl, ExternalEndpointUrl, and WebServiceUrl are grouped in a separate structure named as MIMadditionalHeader. These is due a reason that the value of these fields can change while a message passes through the interoperability systems and therefore are not part of the digital signed message.

XSD Schema (http://mioa.gov.mk/interop/mim/v1**)**

```
<?xmlversion="1.0"encoding="utf-8"?>
<xs:schemaxmlns="http://mioa.gov.mk/interop/mim/v1"xmlns:xs="http://www.w3.org/2001/XM
LSchema"targetNamespace="http://mioa.gov.mk/interop/mim/v1"elementFormDefault="qualifi
ed">

<xs:complexTypename="MIMHeader">
<xs:sequence>
<xs:element minOccurs="1" maxOccurs="1"name="Consumer"type="xs:string">
              <xs:simpleType>
                     <xs:restriction base="xs:string">
                            <xs:maxLength value="50" />
                     </xs:restriction>
              </xs:simpleType>
       </xs:element>
```


This project is funded by the European Union

```
<xs:element minOccurs="0"name="Provider"type="xs:string">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="50" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:element minOccurs="1" maxOccurs="1"name="RoutingToken"type="xs:string">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="100" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:element minOccurs="1" maxOccurs="1"name="Service"type="xs:string">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="100" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:element minOccurs="1" maxOccurs="1"name="ServiceMethod"type="xs:string">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="100" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:element minOccurs="1" maxOccurs="1"name="TransactionId">
                <xs:simpleType>
                        <xs:restrictionbase="xs:string">
                                <xs:patternvalue="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-
        f0-9]{4}-[a-f0-9]{12}"/>
                        </xs:restriction>
                        </xs:simpleType>
        </xs:element>
<xs:element minOccurs="1" maxOccurs="1" name="Dir">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:enumeration value="Request" />
                                <xs:enumeration value="Response" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:elementname="CallType" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                        <xs:restrictionbase="xs:string">
                                <xs:enumerationvalue="synchronous"/>
                                <xs:enumerationvalue="asynchronous"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:elementname="PublicKey"type="xs:string"minOccurs="0"/>

<xs:elementname="MimeType"type="xs:string"minOccurs="0">
        <xs:annotation>
        <xs:documentationxml:lang="en">
        according to http://www.iana.org/assignments/media-types/media-types.xhtml
        </xs:documentation>
        </xs:annotation>
```


This project is funded by the European Union

```
        </xs:element>
<xs:element minOccurs="1" maxOccurs="1"name="TimeStamp"type="xs:dateTime"/>
<xs:elementname="CorrelationId"type="xs:string"minOccurs="0"/>
<xs:element minOccurs="1" maxOccurs="1"name="Signature"type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:schema>

<xs:elementname="MIMadditionalHeader">
<xs:complexType>
<xs:sequence>
<xs:elementname="Status"type="xs:string"minOccurs="0">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="50" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:elementname="StatusMessage"type="xs:string"minOccurs="0">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="255" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>

<xs:elementname="ProviderEndpointUrl"type="xs:string"minOccurs="0">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="255" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>

<xs:elementname="ExternalEndpointUrl"type="xs:string"minOccurs="0">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="255" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
<xs:elementname="WebServiceUrl"type="xs:string"minOccurs="0">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="255" />
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:elementname="CryptoHeader">
<xs:complexType>
<xs:sequence>
<xs:elementname="Key"type="xs:string" minOccurs="1" maxOccurs="1" />
<xs:elementname="InitializationVector"type="xs:string" minOccurs="1" maxOccurs="1" />
<xs:elementname="FormatValue"type="xs:string" minOccurs="1" maxOccurs="1">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
```


This project is funded by the European Union


```
                        <xs:enumeration value="AES" />
                    </xs:restriction>
                </xs:simpleType>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:elementname="MIMBody"id="MIMBody">
<xs:complexType>
<xs:sequence>
<xs:elementname="Message"type="xs:anyType"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```



This project is funded by the European Union

9 MIM Messaging and Configuration

9.1 Principles and Standards

1. MIM service clients and MIM service providers must communicate through a CC.
2. The CC is free to communicate with the CS using an implementation specific protocol.
3. In the case where the MIM is implemented over SOAP, routing will be performed using WS-Addressing.
4. CC will communicate with the CS using https.
5. In order to secure the communication between the communication client and the internal backend systems BS, end-to-end security patterns are advised to be used. Either end-to-end encryption between CC and internal system might be used or at least Client Certificates to assure that only certified and authorized backend systems are allowed to call the CC[10].

9.2 Service access control

Access control is performed by a mapping between organisationswhich are allowed to call which service provided by another organisation.

- Authentication is performed by signing the MIM message using a private certificate assigned to the organisation. Thus the authorized access to provided services can be verified by checking the validity of the signature.
- Authorization is optionally supported by a public key. In this case the SOAP header will contain a public key which contains user information. If present, the IOP system will log the user from the certificate. Authorization however has to happen in the domain of the service provider.

These two mechanisms describe the minimum level of authentication and authorisation which has to be provided by MIM. For internal authentication and authorisation proposes within an interoperability system, additional mechanisms may be supported.

9.2.1 Provider-consumer service access mapping

Access permissions between a service consumer and provider are defined on the CS. How this mapping is defined is outside the scope of MIM but MIM system providers are likely to provide a configuration web interface.

Mapping example:

Consumer	Provider	Service
Institution A	Institution B	*GetCompanies*
Institution A	Institution B	*GetEmployees*
Institution A	Institution C	*GetVehicles*
Institution C	ExtSys1$$InstitutionX	*GetData*

After message reception by the CS, the routing logic on the CS will evaluate the service invocation for validity. If there is no route between consumer and provider configured, the invocation fails, otherwise it is forwarded to the Provider, logging the routing parameters and MIM header parameters.

[10] http://tools.ietf.org/rfc/rfc3280.txt

This project is funded by the European Union

Routing information is stored internally on the CS and indirectly explicated on a CC on which service endpoints are configured, based on reply of GetServices.

For responses to GetServices constructed by a local CS, the RoutingToken consist of the access scheme (usually https) and the communication clients host name, for which the request to GetService was made. The first part of the URL path identifies the providing Communication client, which hosts a service.

Example RoutingToken: cc_min5
Explanation: This information is obtained by a call to GetServices to the CC from Ministry 1. The call will reply the address of the local CC on which services will be implemented as service endpoints and contains further the information that a service getAddress is provided by a CC of Ministry 5 (cc_min5).

ExampleRoutingToken: cc_ext2$$cc_min8
Explanation: This information is obtained by a call to GetServices to the CC from Ministry 1. The call will reply the information of the address of the local CC on which services intended for consumption will be implemented as service endpoints and contains further the information that a service getAddress is provided by a CC cc_ext2. This cc_ext2 is the CC of another interoperability system, where, in case of a service invocation using this RoutingToken, the additional token cc_min8 can be interpreted so that a message can be uniquely identified and routed to the final destination.

9.2.2 Authorization

The MIM architecture relies on the concept of organizational trust, however finer levels of granularity are supported through user authorization. In this case the SOAP header will contain a public key which contains user information. If present, the IOP system will log theuser from the certificate. Authorization however has to happen in the domain of the service provider.

9.3 Providing a service

Providing services according to MIM requires the installation of a CC.

- This CC must be reachable by the CS.
- This CC must be able to physically reach exposed business services.

The provider may register a new service on CS by calling Metaservice*RegisterService* on the CC, whichwill configure that mapping on the CS. The WSDL is published with end points addresses in local environment (provider's environment).

The provider of a MIM CS is free to provide other means of service configuration, however, this is implementation dependent and outside the scope of the MIM specification.

9.4 Service discovery

Service discovery is facilitated by the Metaservices*GetProviders*, *GetServices* and *GetService*.

- *GetProviders* returns a list of identifiers that are providing at least one service to the invoking consumer, including the public key which has to be used for encryption.
- *GetServices* returns all RoutingTokens for certain providers that are available for the invoking consumer.


This project is funded by the European Union

Upon reception of a call to GetServices on the CS, the CS must do the following:

1. Look up a service mapping in the local service access control list (MSC) and
2. Call the same method as which the CS was invoked to one another configurable CC “CCext” in a synchronous manner. If this external call succeeds, information obtained from the local service access control list must be combined with information received from the response to the external CC and the RoutingToken to the calling CC “CCorig” must be assembled as following:

   a) CCorig/cc_min5/Service for information obtained from the local service access control list.
   b) CCorig/CCext/[InstiutionX]/Service for information obtained after a call to GetServices of CCext.
   This information is assembled by the CS upon reception of replies of an external IOP system and will be assembled for further reply to the calling CC. A detailed description of the call sequence, the role of the CS, the local and remote CCs is given in Figure 1: Invocation sequence of MIM MetaserviceGetServices.

- *GetServices*returns all RoutingTokens for certain providers that are available for the invoking consumer
- GetService returns the WSDL for specific provider and service.

The returned WSDL has end point addresses pointing to local CC and therefore invoking those services from consumer perspective is accomplished on the local CC.

This project is funded by the European Union

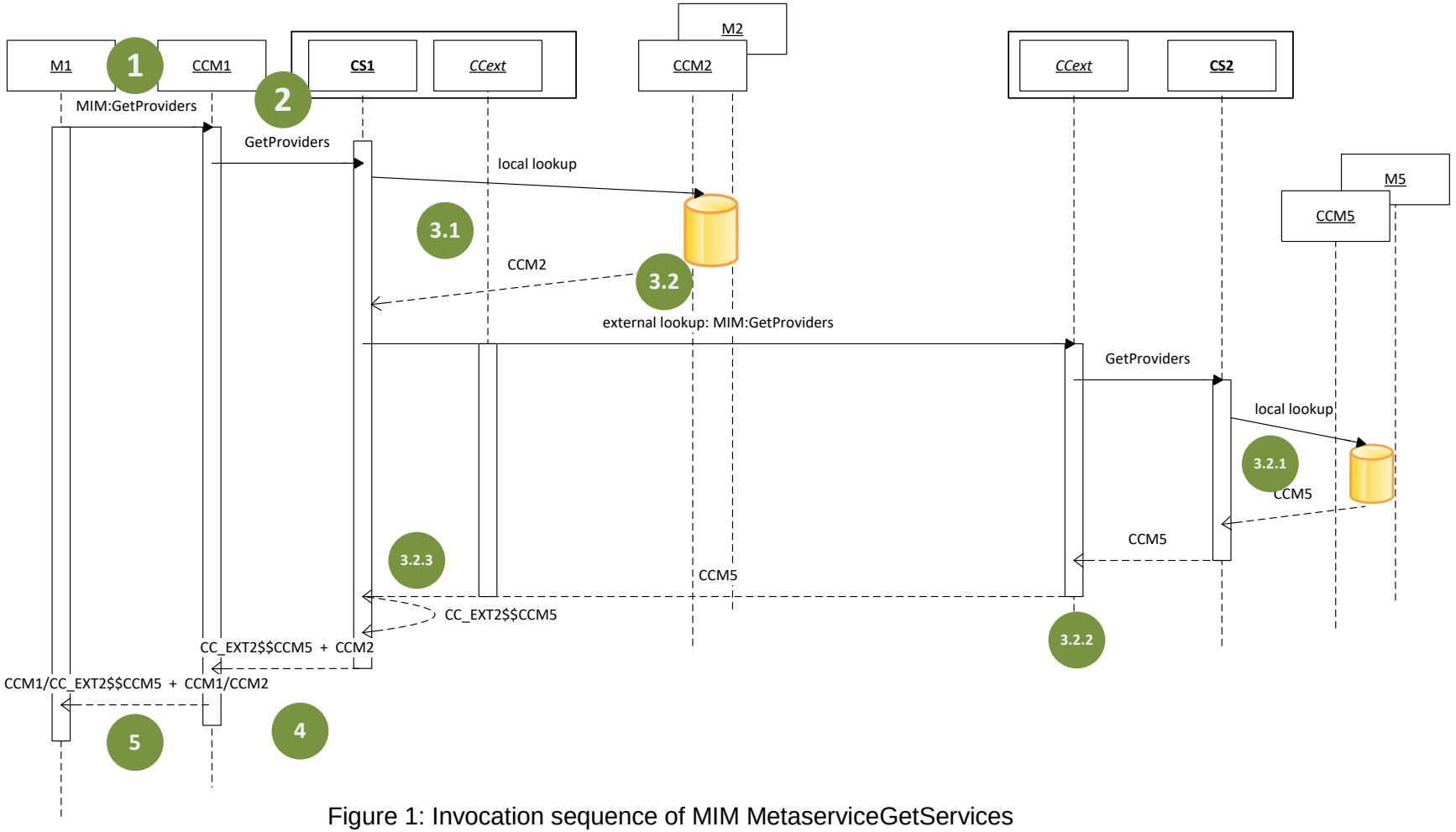

Figure 1: Invocation sequence of MIM MetaserviceGetServices

This project is funded by the European Union

1. Institution M1 is invoking on the local Communication Client CCM1 the Metaservice GetProviders using the MIM Protocol.
2. CCM1 will call the respective method on the local Communication Server CS1.
3.
   3.1. The CS1 will make a lookup in its local data base. In the presented use-case, a service getPerson is provided through CCM2 of institution M2. This service has been configured on the CS1 using the administrative service configuration panel
   3.2. The CS1 will invoke the method GetProviders on its 'external' CC (*CCext*) using the MIM protocol via *CCext* of CS2 (the CS and CCs from one service bus do not communicate directly with the CS or CCs of the other service bus. The communication between the two service buses is done only through the 'external' CCs (*CCext*) in both service buses (one per service bus), thus forming a „bridge" of sorts between the two buses. The *CCext* on each side is a „virtual CC" hosted at the CS (or an application running at the CS) that serves as an interface to the other service bus(es)).
      3.2.1. The 'external' CCext on CS2 will call within its Communication Server CS2 for services M1 is entitled to consume. CS2 must take precautions not to re-invoke the original caller again with a call to GetProviders.
      3.2.2. In this illustrative use-case, the 'external' Communication Client *CCext* of CS2 returns the information, that a Communication Client (CCM5) for institution M5 is providing a service.
      3.2.3.As the CS1 knows the respondent, it prepends the CCext address to the obtained routing information
4. The information of CCext/CCM5 and CCM2 is returned to the invoking Communication Client M1.
5. The obtained information will be used to setup service endpoints by calling GetServices(CCext/CCM5) and GetServices(CCM2) in M1 for getPerson and GetLocation which will point to the local CCM1.

This project is funded by the European Union

9.5 Message Flow

9.5.1 Asynchronous message exchange (Only applicable in MIM level 2)

WSDL with asynchronous request endpoints are returned when Meta service *GetService* is called with parameter *callType*set to *asynchronous.*

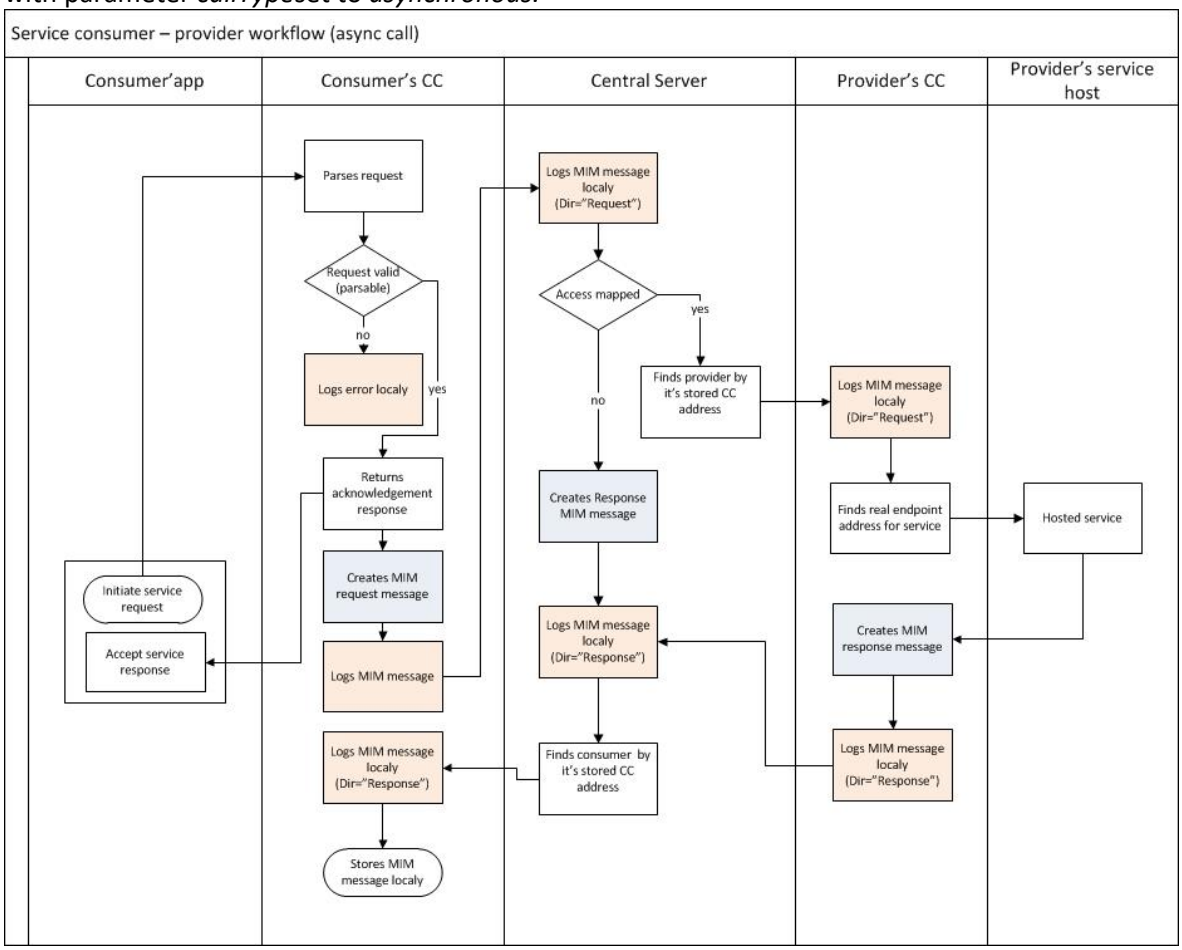

Figure 2 Service consumer – provider workflow of an asynchronous call

When an asynchronous serviceis invoked, the CC returns immediately acknowledging the request with a TransactionId in the SOAP header.
On consumer's CC all messages from asynchronous calls are collected and stored in MIM format. Collected messages can be consumed by MetaservicesGetMessage and GetMessageByTransactionId. These MIM services are typically called by a backend system BS.

9.5.2 Synchronous message exchange between institutions

This project is funded by the European Union

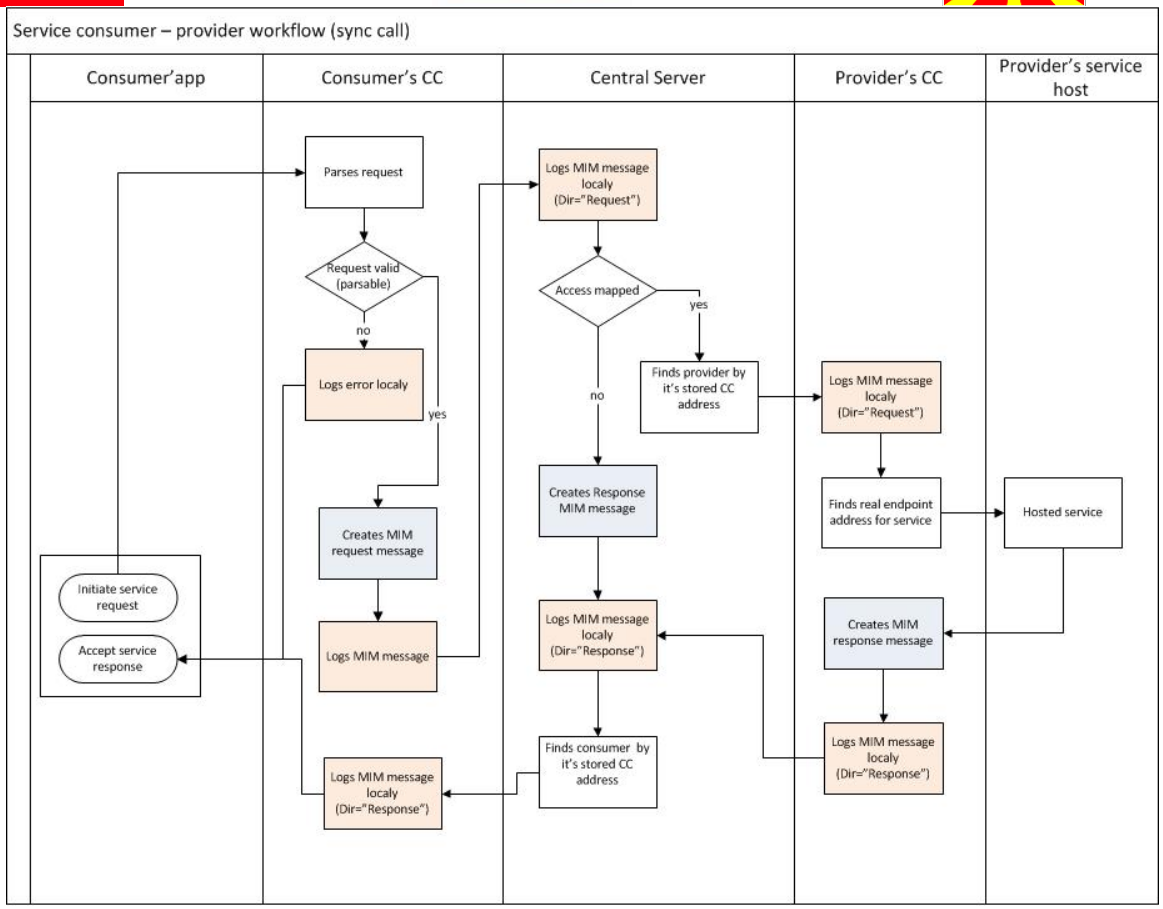

Figure 3 Service consumer – provider workflow of a synchronous call

9.5.3 Use case: Institution A intends to invoke a service method getCompaniesprovided by institution B.

Application hosted in institution A invokes service *CompanyService/getCompanies*on its own CC (the WSDL was received by *GetService*metaservice). The service is invoked in the format

https://institutionA/institutionB/CompanyService/getCompanies

withfirst segment is the URLof the local CC host name, (second segment the RoutingToken containing at least the local destination of the recipients CC, third segment is the ServiceId - are optional).
The SOAP message is posted as a HTTP request body to that CC endpoint.

The CC possibly encrypts the posted XML message and creates MIM message structure where it adds

- Consumer – Identifier of the institution as obtained from the OPS
- Service – ServiceId as registered in the MSC
- publicKey – optional
- Message body of actual request if required
- TransactionId
- Dir – for MIM level 1 this will always be "Request"
- Timestamp

This project is funded by the European Union

as well as any other MIM header parameter defined as mandatory. The signature is calculated and put into the appropriate MIM header field.

MIM message is forwarded to the CS which logs the request.

The CS holds information about service access rights.

- If the invocation violates existing access rights from institution A to institution B for service CompanyService/getCompaniesthe attempt is logged as invalid and a response is generated with MIM Status field set to 403, message body containing standard SOAP exception and optionally further information set in the MIM header StatusMessage.
- If the invocation does not violate access rights, the CS invokes the service over the SOAP protocol on the CC of institution B. CC on institution B receives message from MIM SOAP message body, checks signature, decrypts the message body and posts message to internal system that hosts the service.

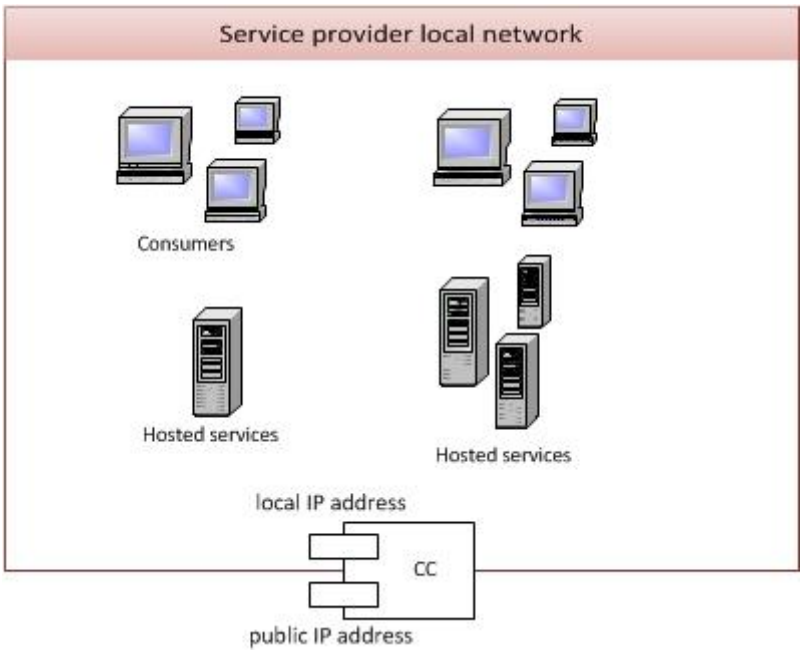

Figure 4. Communication client from Provider's perspective

Seen from the institution, the CC is the single only point to communicate with the CS. The CC in turn has direct access to all backend services which either provide or consume services on the MIM system.

9.5.4 Use case: Integration with external systems

Service provision and invocation across MIM systems is provided by the fact that every CS has to listen to metaservices as described above by proving a CC endpoint.

This project is funded by the European Union

All services provided by an external MIM system will be available to participants in a local MIM platform through a RoutingToken[11] that uniquely identifies the remote CS which will check for a valid service invocation mapping and forward the message to the final consuming or providing CC.

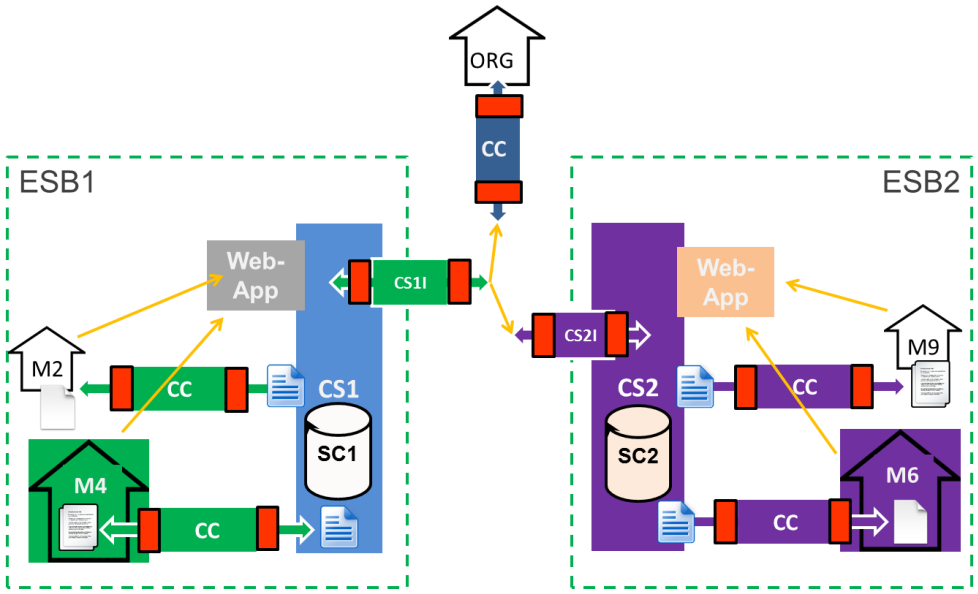

Figure 5: Communication across MIM systems

Informative: The CC as service consumer and service provider

In the role of a service consumer the CC hosts WSDL documents of other service providers so applications from local CC network may consume them. All WSDL service endpoints are pointing to the CC itself.

The WSDL URLs are formed as following: https://Consumer/RoutingToken/ServiceId

1. CC handles all requests for its domain and parses them in order to resolve the destination CC. The RoutingToken consists of the ProviderID or in cases where the provider is outside local MIM system, it denotes the CC of the destinations CS and contains further routing information which will be passed via the CC of the external IOP system which the external CS will use to resolve the path to the final CC in its domain.
2. CC generates *TransactionId*
3. CC logs information from the MIM header
4. CC encrypts received SOAP message (request body) with final destinations CC public key.
5. CC creates / appends to MIM message with following data:

- Consumer – Identifier of the institution as obtained from the OPS
- Service – ServiceId as registered in the MSC
- publicKey – optional
- Message body of actual request if required
- TransactionId

[11]This RoutingToken is obtained by a call to GetProviders.

This project is funded by the European Union

- Dir – for MIM level 1 this will always be “Request”
- Timestamp
- CC calculates Signature and puts it into the appropriate MIM header field

6. CC forwards message to CS.
7. CC waits for response and forwards that response to sender of request (from point 1).

In the case where the CC has stored on itself original endpoint to provide a service from its own network, an invocation as a service provider has the following information flow:

1. CC checks signature of received message and decrypts the message body.
2. CC logs parameters from the MIM header
3. CC gets original endpoint from local store for extracted service.
4. CC sends POST request to original endpoint.
5. Response is encrypted with consumer’s CC public key
6. When the application successfully responded, the CC creates a new MIM message and copies the original headers request parameters into it, adds the mime type of the response,changes Dir to “Response”and encryptsthe message response. Then the CC returns the created MIM message to the CS.