# OMA Interface

**Introduction**

The Open Mobile Alliance Interface (OMA Interface) is an API of ALTAMIDES that follows Mobile Location Protocol (MLP). MLP is an application-level protocol for getting the position of mobile stations (mobile phones, wireless personal digital assistants, etc.) independent of underlying network technology, i.e. independent of location derivation technology, as defined by the Open Mobile Alliance. The MLP serves as the interface between a Location Server (ALTAMIDES) and an MLS Client.

## **SDD**

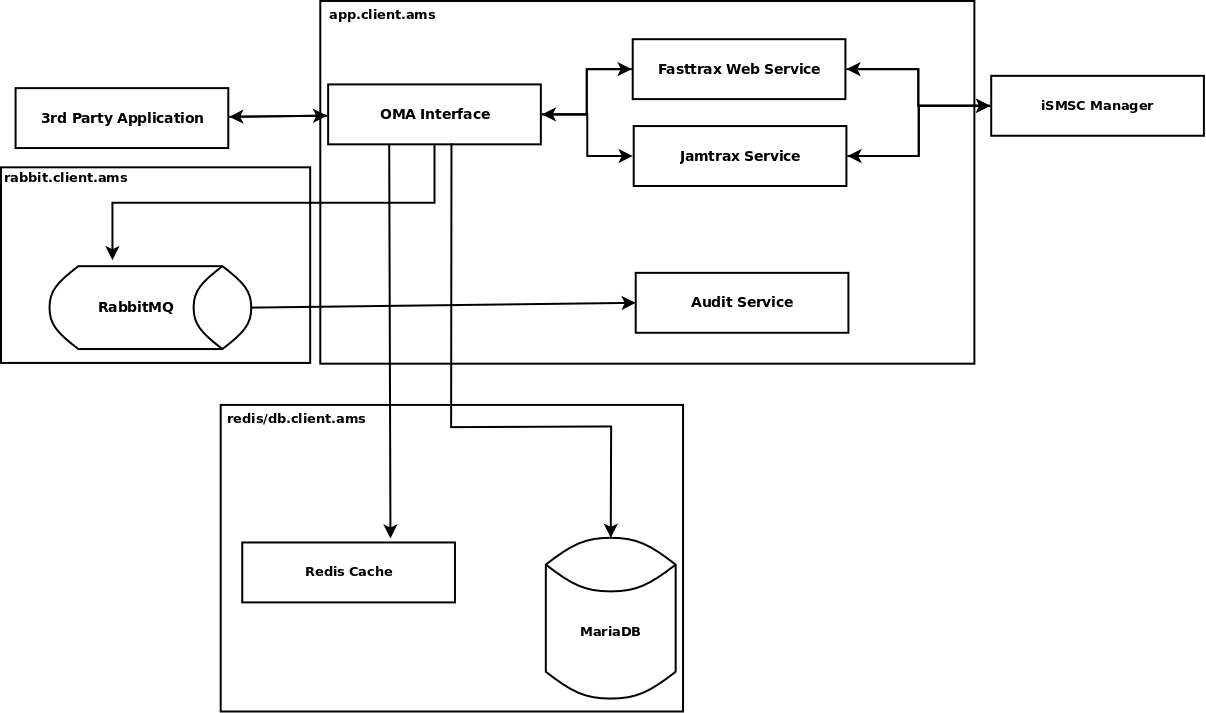
References to the different SDD files in Alfresco related to this module.

* SDD: <https://doc-srv.1rstwap.com/share/page/site/development/document-details?nodeRef=workspace://SpacesStore/2b4274b7-91ff-47f6-8a18-00be37b3d606>

## **User Manual**

* User Manual: [OMA Interface Documentation](https://doc-srv.1rstwap.com/share/page/site/development/document-details?nodeRef=workspace://SpacesStore/e094b7ba-1d45-446f-b1c2-2898ed6fe0e3)

## **Architecture Diagram**



## **OMA Interface Table**

For interrogation requests in SLIR, TLRR, and DLIR the Location Request ID and Error Code   
returned by the FastTrax WS needs to be stored as well. Because for PJSR we don’t do an   
interrogation request, that means the LOCATION\_REQUEST\_ID and   
LOCATION\_RESULT\_CODE is nullable.  
We store the data and time (in GMT), the IP Address and the User ID that was matched for   
the incoming request. We also store the targeted number for each request. For services such   
as TLRS and DLIS we store the URL which needs to be called to forward each interrogation   
result.

## **Services**

The following services are implemented in ALTAMIDES:

|  |  |  |  |
| --- | --- | --- | --- |
| Service | Short | Status | Messages |
| Standard Location Immediate Service | SLIS | Basic Implemented |  |
| SLIR |
| SLIA |
| SLIREP |
| Detailed Location Immediate Service | DLIS | 1rstWAP special |  |
| DLIR |
| DLIA |
| DLIREP |
| Triggered Location Reporting Service | TLRS | Implemented |  |
| TLRR |
| TLRA |
| TLREP |
| TLRSR |
| TLRSA |
| Passive Jamming Reporting Service | PJRS | 1rstWAP special |  |
| PJRR |
| PJRA |
| PJREP |
| PJRSR |
| PJRSA |

## **OMA Mobile Location Protocol (MLP)**

This document describes all the services which are currently available in the ALTAMIDES. All services use Mobile Location Protocol (MLP) version 3.3 as defined in the Open Mobile Alliance documentation. It is basically an XML format for sending requests and responses. The typical usage for requests are:

<?xml version="1.0"?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_INIT\_330.DTD">

<svc\_init ver="3.3.0">

<hdr ver="3.3.0">

<client>

<id>locmo</id>

<pwd>rSdNpCSTvMAc79DeD0EuL0lCffhQWYRX</pwd>

</client>

</hdr>

{...}

</svc\_init>

The hdr element contains authentication information of the ALTAMIDES (web) user account that sends the request. The id is the web access username of the account and pwd is the encrypted password of that account. The password can be retrieved from the ALTAMIDES database, herefore you would need to get support from 1rstWAP or one of our partners. The {...} should be replaced with request XML element, see a request DTD in the other chapters for more information. The typical usage for responses are:

<?xml version="1.0"?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

{...}

</svc\_result>

The {...} will be replaced by ALTAMIDES with the appropriate response XML element.

### **Standard Location Immediate Service - SLIS**

SLIR is a request message for requesting the location of one Mobile Subscriber (MS) or named Single Interrogation. We provide 3 types of MS, those are of Phone number, MSISDN and IMSI.

#### Standard Location Immediate Request (SLIR) example in complete format:

<?xml version="1.0"?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_INIT\_330.DTD">

<svc\_init ver="3.3.0">

<hdr ver="3.3.0">

<client>

<id>locmo</id>

<pwd>rSdNpCSTvMAc79DeD0EuL0lCffhQWYRX</pwd>

</client>

</hdr>

<slir ver="3.3.0" res\_type="ASYNC">

<msid type="msisdn">9711234567894</msid>

<client\_req\_id>OMA\_TEST</client\_req\_id>

<haQueryMethod>samoussa</haQueryMethod>

<pushaddr>

<url>http://10.32.6.18:4202/test/echo</url>

</pushaddr>

</slir>

</svc\_init>

Possible MSID types are:

* PHONE NUMBER
* MSISDN
* IMSI

And client\_req\_id is alphanumeric, this element can be used for your own system to keep track of your own requests.  
Optional parameters are haQueryMethod and pushaddr, these are required for high accuracy interrogation and res\_type need to be changed to "ASYNC" for this request.

### Standard Location Immediate Answer (SLIA)

Standard Location Immediate Answer (SLIA) is MLP formatted response for SLIR which contains location information of the requested Mobile Subscriber number.

#### Standard Location Immediate Answer (SLIA) example in complete format (success):

<?xml version="1.0"?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result>

<slia ver="3.3.0">

<pos>

<msid>62123456789</msid>

<pd>

<time utc\_off="+0000">20141221115009</time>

<shape>

<CircularArea>

<coord>

<X>6 14 26.52 S</X>

<Y>106 49 10.93 E</Y>

</coord>

<radius>0</radius>

</CircularArea>

</shape>

</pd>

</pos>

<client\_req\_id>12345</client\_req\_id>

</slia>

</svc\_result>

#### Standard Location Immediate Answer (SLIA) example in complete format (failed):

<?xml version="1.0"?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result>

<slia ver="3.3.0">

<pos>

<poserr>

<result resid="3">UNAUTHORIZED APPLICATION</result>

<add\_info>Password not matched.</add\_info>

<time utc\_off="+0000">20141217184146</time>

</poserr>

</pos>

<client\_req\_id>12345</client\_req\_id>

</slia>

</svc\_result>

### Standard Location Immediate Report (SLIREP)

Standard Location Immediate Report (SLIREP) is MLP formatted response for SLIR which contains location information of the requested Mobile Subscriber number when a High Accuracy and ASYNC res\_type is requested via the SLIR.

#### Standard Location Immediate Report (SLIREP) example in complete format:

<!DOCTYPE svc\_result PUBLIC "-//OMA//DTD MLP 3.3.0//EN" "http://www.openmobilealliance.org/Tech/dtd/MLP\_SVC\_RESULT\_330.DTD">

<svc\_result>

<slirep ver="3.3.0">

<pos>

<msid>9711234567894</msid>

<pd>

<interrogationDetails>

<interrogationHighAccuracyMethod>oss</interrogationHighAccuracyMethod>

</interrogationDetails>

<locationDetails>

<highAccuracy>

<highAccuracyShape>Polygon</highAccuracyShape>

<includedAngle>0</includedAngle>

<innerRadius>0</innerRadius>

<latitude>30.083552402942804</latitude>

<longitude>31.450043839507305</longitude>

<offsetAngle>0</offsetAngle>

<polygon>

<numberOfPoints>0</numberOfPoints>

</polygon>

<uncertaintyRadius>0.000000</uncertaintyRadius>

</highAccuracy>

</locationDetails>

</pd>

</pos>

</slirep>

</svc\_result>

### Detailed Location Immediate Request (DLIR)

The Detailed Location Immediate Service or DLIS is an advanced version of SLIS complete format.

#### Detailed Location Immediate Request (SLIR) example in complete format:

<?xml version="1.0"?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_INIT\_330.DTD">

<svc\_init ver="3.3.0">

<hdr ver="3.3.0">

<client>

<id>annissa</id>

<pwd>cKOvUE35HT9vEa1O1aiX++Awbs7ycsDm</pwd>

</client>

</hdr>

<dlir ver="3.3.0" res\_type="SYNC">

<msid type="MSISDN">6281281130320</msid>

<client\_req\_id>12345</client\_req\_id>

<node>Automatic</node>

<queryMethod>Automatic</queryMethod>

<haQueryMethod>OSS</haQueryMethod>

<pushaddr>

<url>http://mysystem.com:1234/callbackUrl</url>

</pushaddr>

</dlir>

</svc\_init>

Possible MSID types are:

* PHONE NUMBER
* MSISDN
* IMSI
* IMEI

For Node and queryMethod can use "Automatic" or "AUT"

And client\_req\_id is alphanumeric, this element can be used for your own system to keep track of your own requests.

Optional parameters are haQueryMethod and pushaddr, these are required for high accuracy interrogation and res\_type need to be changed to "ASYNC" for this request.

### Detailed Location Immediate Answer (DLIA)

Detailed Location Immediate Answer (DLIA) is MLP formatted response for DLIR which contains location information of the requested Mobile Subscriber number.

#### Detailed Location Immediate Answer (DLIA) example in complete format (success):

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

<!DOCTYPE svc\_result PUBLIC "-//OMA//DTD MLP 3.3.0//EN" "http://www.openmobilealliance.org/Tech/dtd/MLP\_SVC\_RESULT\_330.DTD">

<svc\_result>

<dlia ver="3.3.0">

<pos>

<msid>6281281130320</msid>

<pd>

<time utc\_off="+0000">20180719043317</time>

<shape>

<CircularArea>

<coord>

<X>3 8 10.07 N</X>

<Y>101 41 10.80 E</Y>

</coord>

</CircularArea>

</shape>

<interrogationDetails>

<interrogationDatetime>2018-07-19 04:33:17</interrogationDatetime>

<interrogationMethod>ATI</interrogationMethod>

<interrogationNode>ORO</interrogationNode>

<interrogationResult>Target Interrogation Successful</interrogationResult>

</interrogationDetails>

<subscriberDetails>

<ageOfLocation>0</ageOfLocation>

<forwardingNumber>Not Available</forwardingNumber>

<forwardingStatus>When Unreachable</forwardingStatus>

<imei>357325071400238</imei>

<imsi>Not Available</imsi>

<phoneStatus>Idle</phoneStatus>

</subscriberDetails>

<locationDetails>

<cell>

<azimuth>-1</azimuth>

<btsLocation>

<latitude>Not Available</latitude>

<longitude>Not Available</longitude>

</btsLocation>

<cellCentreLocation>

<latitude>3.13613000</latitude>

<longitude>101.68633000</longitude>

</cellCentreLocation>

<cellQuality>Estimated Cell Area</cellQuality>

<cellReference>502.19.1243.36001</cellReference>

<locationAddress>No vicinity found</locationAddress>

<radius>536</radius>

</cell>

</locationDetails>

<telcoData>

<additionalNetInfo>Not Available</additionalNetInfo>

<homeCountry>Indonesia</homeCountry>

<homeOperator />

<netInServiceSince>Not Available</netInServiceSince>

<roamingCountry>Malaysia</roamingCountry>

<roamingNumber>Not Available</roamingNumber>

<roamingOperator>Celcom</roamingOperator>

<supportedTechnology>Not Available</supportedTechnology>

<usedMethod>ATI</usedMethod>

<usedNode>ORO</usedNode>

<vlrGlobalName>MYS01</vlrGlobalName>

<vlrGlobalTitle>60123456789</vlrGlobalTitle>

</telcoData>

</pd>

</pos>

<client\_req\_id>12345</client\_req\_id>

</dlia>

</svc\_result>

#### Dtandard Location Immediate Answer (DLIA) example in complete format (failed):

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

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result>

<dlia ver="3.3.0">

<pos>

<poserr>

<result resid="3">UNAUTHORIZED APPLICATION</result>

<add\_info>Password not matched.</add\_info>

<time utc\_off="+0000">20141217184146</time>

</poserr>

</pos>

<client\_req\_id>12345</client\_req\_id>

</dlia>

</svc\_result>

### Detailed Location Immediate Report (DLIREP)

The Detailed Location Immediate Report (DLIREP) s a response for an asynchronous DLIR and it will return information related to the high accuracy-based location interrogation result if the res\_type in DLIR was set to ASYNC.  
Detailed Location Immediate Report (DLIREP) example in complete format:

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

<!DOCTYPE svc\_result PUBLIC "-//OMA//DTD MLP 3.3.0//EN"

"http://www.openmobilealliance.org/Tech/dtd/MLP\_SVC\_RESULT\_330.DTD">

<svc\_result>

<dlirep ver="3.3.0">

<pos>

<msid>6281374041104</msid>

<interrogationDetails>

<interrogationDatetime>2017-11-14 04:41:00</interrogationDatetime>

<interrogationNode>OR</interrogationNode>

<interrogationResult>Target Interrogation Successful

</interrogationResult>

<interrogationHighAccuracyMethod>OSS

</interrogationHighAccuracyMethod>

</interrogationDetails>

<locationDetails>

<highAccuracy>

<highAccuracyShape>Point with Circle</highAccuracyShape>

<latitude>-7.17352875</latitude>

<longitude>107.72538362</longitude>

<uncertaintyRadius>50</uncertaintyRadius>

</highAccuracy>

</locationDetails>

</pos>

</dlirep>

</svc\_result>

### Triggered Location Reporting Service - TLRS

The Triggered Location Reporting Service (TLRS) requests for a mobile subscriber’s location to be reported on a regular interval.

#### Triggered Location Reporting Request (TLRR) example in complete format:

<?xml version="1.0"?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_INIT\_330.DTD">

<svc\_init ver="3.3.0">

<hdr ver="3.3.0">

<client>

<id>locmo</id>

<pwd>rSdNpCSTvMAc79DeD0EuL0lCffhQWYRX</pwd>

</client>

</hdr>

<tlrr ver="3.3.0">

<msid type="MSISDN">628997985184</msid>

<start\_time>20150319163600</start\_time>

<stop\_time>20150319163700</stop\_time>

<interval>10000</interval>

<haQueryMethod>samoussa</haQueryMethod>

<pushaddr>

<url>http://mysystem:1234/callbackUrl</url>

</pushaddr>

<client\_req\_id>abc123</client\_req\_id>

</tlrr>

</svc\_init>

Parameter <haQueryMethod> is optional if user want to include high accuracy interrogation result in TLREP.  
The following rules apply to the use of ‘start\_time’, ‘stop\_time’, ‘interval’:

* If no START\_TIME is specified reporting SHALL start immediately.
* If no STOP\_TIME is specified the reporting SHOULD occur until explicitly cancelled with ‘Triggered Location Reporting Stop Request (TLRSR)’ or a time out occurs (depending on system configuration). Timeout MAY be reported to the LCS client by 'time\_remaining' in triggered location report.
* If STOP\_TIME is ‘older’ than then current time then the Location Server MUST reject the request with an error indication ‘110’ to the client.
* If STOP\_TIME is earlier than START\_TIME then the implementation MUST reject the request with an error indication ‘110’ to the client.
* If STOP\_TIME is equal to START\_TIME then the Location Server MUST return a single location report to the client at the specified time. Any interval specified MUST be ignored.
* The INTERVAL defines how often a TLRR will be send in milliseconds.

Possible MSID types are:

* PHONE NUMBER
* MSISDN
* IMSI

### Triggered Location Reporting Answer (TLRA)

The Triggered Location Reporting Answer (TLRA) is the response to a TLRR that acknowledges the request.

#### Triggered Location Reporting Answer (TLRA) example in complete format (success):

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<tlra ver="3.3.0">

<client\_req\_id>abc123</client\_req\_id>

<req\_id>a0c71fc9-502c-49a2-8221-a6303f68a874</req\_id>

</tlra>

</svc\_result>

#### Triggered Location Reporting Answer (TLRA) example in complete format (failed):

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<tlra ver="3.3.0">

<client\_req\_id>abc123</client\_req\_id>

<result resid="4">UNKNOWN SUBSCRIBER</result>

</tlra>

</svc\_result>

### Triggered Location Report (TLREP)

The Triggered Location Report (TLREP) are location interrogation responses following the specification (interval, start time, stop time, and push address) of the TLRR.

#### Triggered Location Report (TLREP) example in complete format:

<?xml version="1.0" ?>

<!DOCTYPE svc\_result PUBLIC "-//OMA//DTD MLP 3.3.0//EN" "http://www.openmobilealliance.org/Tech/dtd/MLP\_SVC\_RESULT\_330.DTD">

<svc\_result>

<tlrep ver="3.3.0">

<trl\_pos>

<msid>9711234567894</msid>

<pd>

<time utc\_off="+0000">20191003041537</time>

<shape>

<CircularArea>

<coord>

<X>6 11 14.03 S</X>

<Y>106 50 29.07 E</Y>

</coord>

<radius>734</radius>

</CircularArea>

</shape>

<interrogationDetails>

<interrogationHighAccuracyMethod>etisalat</interrogationHighAccuracyMethod>

</interrogationDetails>

<locationDetails>

<highAccuracy>

<highAccuracyShape>Arc</highAccuracyShape>

<includedAngle>132</includedAngle>

<innerRadius>200</innerRadius>

<latitude>25.256200000000000</latitude>

<longitude>55.312900000000000</longitude>

<offsetAngle>350</offsetAngle>

<uncertaintyRadius>71.000000</uncertaintyRadius>

</highAccuracy>

</locationDetails>

</pd>

</trl\_pos>

</tlrep>

</svc\_result>

Parameter <locationDetails> is optional and only be sent if high accuracy is requested in TLRR.

### Triggered Location Reporting Stop Request (TLRSR)

The Triggered Location Reporting Stop Request (TLRSR) will request to stop triggered location reporting that have been requested by a TLRR before. It will be responded by an TLRSA.

#### Triggered Location Reporting Stop Request (TLRSR) example in complete format:

<?xml version="1.0" ?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_INIT\_330.DTD">

<svc\_init ver="3.3.0">

<hdr ver="3.3.0">

<client>

<id>locmo</id>

<pwd>rSdNpCSTvMAc79DeD0EuL0lCffhQWYRX</pwd>

</client>

</hdr>

<tlrsr ver="3.3.0">

<msid type="MSISDN">628997985184</msid>

<req\_id>a0c71fc9-502c-49a2-8221-a6303f68a874</req\_id>

<client\_req\_id>def678</client\_req\_id>

</tlrsr>

</svc\_init>

### Triggered Location Reporting Stop Answer (TLRSA)

Triggered Location Reporting Stop Answer (TLRSA) is a response to a TLRSR.

#### Triggered Location Reporting Stop Answer (TLRSA) example in complete format (success):

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<tlrsa ver="3.3.0">

<msid type="MSISDN">628997985156</msid>

<req\_id>a0c71fc9-502c-49a2-8221-a6303f68a874</req\_id>

<client\_req\_id>def678</client\_req\_id>

</tlrsa>

</svc\_result>

#### Triggered Location Reporting Stop Answer (TLRSA) example in complete format (failed):

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<tlrsa ver="3.3.0">

<result resid="115">INVALID MSID IN TLRSR</result>

<client\_req\_id>def678</client\_req\_id>

</tlrsa>

</svc\_result>

### Passive Jamming Reporting Service - PJRS

The Triggered Location Reporting Service (TLRS) requests for a mobile subscriber’s location to be reported on a regular interval.

#### Passive Jamming Reporting Request (PJRR) example in complete format:

<?xml version="1.0" ?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_INIT\_330.DTD">

<svc\_init ver="3.3.0">

<hdr ver="3.3.0">

<client>

<id>locmo</id>

<pwd>rSdNpCSTvMAc79DeD0EuL0lCffhQWYRX</pwd>

</client>

</hdr>

<pjrr ver="3.3.0">

<msid type="MSISDN">628997985184</msid>

<pjrr\_service>all</pjrr\_service>

<start\_time>20150319163600</start\_time>

<stop\_time>20150319163700</stop\_time>

<pushaddr>

<url>http://mysystem:1234/callbackUrl</url>

</pushaddr>

<client\_req\_id>klm789</client\_req\_id>

</pjrr>

</svc\_init>

The following rules apply to the use of ‘start\_time’, ‘stop\_time’, ‘interval’:

* pjrr\_service options are: gsm, data, and all:

gsm: passive jam incoming SMS and incoming voice calls to the targeted MS

data: passive jam incoming data communication to and from the targeted MS

all: passive jam both SMS, voice calls, and data.

* If no START\_TIME is specified reporting SHALL start immediately.
* If no STOP\_TIME is specified the jamming SHOULD occur until explicitly cancelled with Passive Jamming Stop Request (PJSR) or by default the mobile subscriber will be unjammed after 24 hours.
* If STOP\_TIME is in the past then the Passive Jamming Reporting Service MUST reject the request with an error indication ‘110’ to the client.
* If STOP\_TIME is earlier than START\_TIME then the Passive Jamming Reporting Service MUST reject the request with an error indication ‘110’ to the client.
* If STOP\_TIME is equal to START\_TIME then the Passive Jamming Server MUST return a single passive jamming report to the client at the specified time. Any interval specified MUST be ignored.

Possible MSID types are:

* PHONE NUMBER
* MSISDN
* IMSI

### Triggered Location Reporting Answer (TLRA)

The Triggered Location Reporting Answer (TLRA) is the response to a TLRR that acknowledges the request.

#### Passive Jamming Reporting Answer (PJRA) example in complete format (success):

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<pjra ver="3.3.0">

<req\_id>a0c71fc9-502c-49a2-8221-a6303f68a874</req\_id>

<client\_req\_id>klm789</client\_req\_id>

</pjra>

</svc\_result>

#### Passive Jamming Report (PJREP)

The Passive Jamming Report (PJREP) is a response that confirms that ALTAMIDES is attempting to jam the targeted mobile subscriber. This report will only be sent once and will be sent after a successful PJRR.

#### Passive Jamming Report (PJREP) example in complete format:

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<pjrep ver="3.3.0">

<client\_req\_id>AADDZZ1721403322</client\_req\_id>

<req\_id>a0c71fc9-502c-49a2-8221-a6303f68a874</req\_id>

<rpj\_pos>

<msid type="MSISDN">628997985184</msid>

<pd>

<time utc\_off="+0000">20150326095942</time>

</pd>

</rpj\_pos>

</pjrep>

</svc\_result>

#### Passive Jamming Reporting Stop Request (PJRSR)

The Passive Jamming Reporting Stop Request (PJRSR) is used to unjam a Mobile Subscriber (MS). It will be followed up by a PJRSA response.

#### Passive Jamming Reporting Stop Request (PJRSR) example in complete format:

<?xml version="1.0" ?>

<!DOCTYPE svc\_init SYSTEM "MLP\_SVC\_INIT\_330.DTD">

<svc\_init ver="3.3.0">

<hdr ver="3.3.0">

<client>

<id>locmo</id>

<pwd>rSdNpCSTvMAc79DeD0EuL0lCffhQWYRX</pwd>

</client>

</hdr>

<pjrsr ver="3.3.0">

<msid>62987654321</msid>

<req\_id>a0c71fc9-502c-49a2-8221-a6303f68a874</req\_id>

<client\_req\_id>rst345</client\_req\_id>

</pjrsr>

</svc\_init>

#### Passive Jamming Reporting Stop Answer (PJRSA)

The Passive Jamming Reporting Stop Answer (PJRSA) is a response to a PJRSR. It confirms whether the stop request for the jammed Mobile Subscriber (MS) is acknowledged.

#### Passive Jamming Reporting Stop Answer (PJRSA) example in complete format (success):

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<pjrsa ver="3.3.0">

<msid type="MSISDN">62987654321</msid>

<req\_id>a0c71fc9-502c-49a2-8221-a6303f68a874</req\_id>

<client\_req\_id>rst345</client\_req\_id>

</pjrsa>

</svc\_result>

#### Triggered Location Reporting Stop Answer (TLRSA) example in complete format (failed):

<?xml version="1.0" ?>

<!DOCTYPE svc\_result SYSTEM "MLP\_SVC\_RESULT\_330.DTD">

<svc\_result ver="3.3.0">

<tlrsa ver="3.3.0">

<result resid="115">INVALID MSID IN TLRSR</result>

<client\_req\_id>def678</client\_req\_id>

</tlrsa>

</svc\_result>

# JAMTRAX

# **Introduce**

JamTrax is one of ALTAMIDES modules. Under this module, GSM devices of pre-registered targets (ProfileTrax) and non-registered targets (ad-hoc) can be jammed via various GSM network identifiers, such as MSISDN, IMSI, and IMEI. This jamming process allows the user to jam the incoming SMS and incoming phone calls (called Passive Jamming) and data communication can be jammed as well (Data Jamming).

## **SDD**

References to the different SDD files in Alfresco related to this module.

* SDD: https://doc-srv.1rstwap.com/share/page/site/development/document-details?nodeRef=workspace://SpacesStore/5e3010cc-a626-42b9-82ce-2c62ebc85956

**How the Jam Service Works**

The Jam Service has 2 different flow for processing jamming requests, scheduled jamming and on-demand jamming. On-demand jamming (not scheduled) is supported per target (as defined in ProfileTrax) or ad-hoc. Scheduled jamming has a time interval to check the schedule of a target frequently from the database. It checks which target should be jammed or unjammed, hence this service has defined in its configuration a cron job. Passive Jam and Data Jam are requested directly from Jamming Service to the iSMSC Manager’s Jam Router (Jamming API). Following are steps which are executed when the cron job of Jam Service is called;

1. Retrieves from database all targets which needs to be Jammed or Unjammed from table RUNNING\_SCHEDULE in database LOCATION\_SINGLE\_USE,
2. START\_DATE\_SERVER and END\_DATE\_SERVER column in table RUNING\_SCHEDULE in database LOCATION\_SINGLE\_USE will be checked to determine if the target is in the correct time for jamming,
3. Once a target has been jammed successfully, table RUNNING\_SCHEDULE will be updated with value column lock = 1 and column server\_name with value of the server name and run\_schedule will be updated with current date and time,
4. Jamming service will notify user who is assigned for that target via Websocket Service,
5. If the target from process B is not in the correct time and current status of target is jammed, the target will be unjammed,
6. A target will be removed from the jamming list straightly from database once a target is unjammed successfully,
7. Jamming service will notify user who is assigned for that target via Websocket Service,
8. Since the cron job will do above steps in each interval, then the maintenance of Jam or Unjam status is also the responsibility of the cron job.

Input of Jam Service is solely from Database Schedule, therefore the usage of Jam Service is only for internal process. Although, Jam Service queue is exists in RabbitMQ. The expired or inactive Jamming schedule will be removed from the database immediately by the following steps;

1. Jamming Schedule that is deleted from the side of User Interface (UI) will be marked as deleted. Afterwards, Jam Service will notice the target as an unjammed target.
2. Jam Service will remove it from database when it is unjammed successfully.

Target jamming and Ad-Hoc Jamming have more simple steps since they do not need to use cron job but instead send the request directly to iSMSC-Manager after receiving jamming request from front-end side.

Following are steps which are executed when user request for Target or Ad hoc jamming :

1. Jamming service receive jamming request from UI via RabbitMQ.
2. Target data will be inserted to the RUNNING\_SCHEDULE, JAMMING\_SCHEDULE, JAMMING\_SCHEDULE\_TARGET, JAMMING\_TASK\_CACHE table in database LOCATION\_SINGLE\_USE.
3. Jamming service will send request to iSMSC-Manager to jam or unjam the target.
4. Jamming service will update RUNNING\_SCHEDULE, JAMMING\_SCHEDULE\_HISTORY, JAMMING\_SCHEDULE, JAMMING\_SCHEDULE\_HISTORY\_TARGET, JAMMING\_SCHEDULE\_TARGET, JAMMING\_TASK\_CACHE tables in database LOCATION\_SINGLE\_USE based on response from iSMSC-Manager.
5. If the request is unjam or the iSMSC-Manager send an error response, schedule data for the target will be deleted.
6. Jamming service send back the result to the UI via Websocket Service.

As runs in the FastTrax Scheduler, by using a row locking in database, Jamming Service likewise accommodate running on a multiple instance for high availability solution.

### Sequence Diagram of Jam Service

The sequence of Jam Service for scheduled jamming is divided into four parts of process:

1. Requested method of jamming process
2. Scheduler process
3. Cron job process
4. The callback

When the user is request to jam a target number, the GUI will send the request according to the selected method to the Routing Service via RabbitMQ, the method is either Jam, Unjam, or UnjamAll. Please note that all requests and responses will be inserted from RabbitMQ to the services. After Routing Service get the request from UI, this service will continue sending the request to JamTrax Service. Afterwards, by passing through the Routing Service, the values of request will be stored into the database via DB Service. When the insertion is true, JamTrax Service will get the response from DB Service through the Router Service then send the response to PHP as acknowledge and display the notification to UI in the browser as waiting to Jam/Unjam. Otherwise, the information of acknowledge will be failed and will be aborted the process.

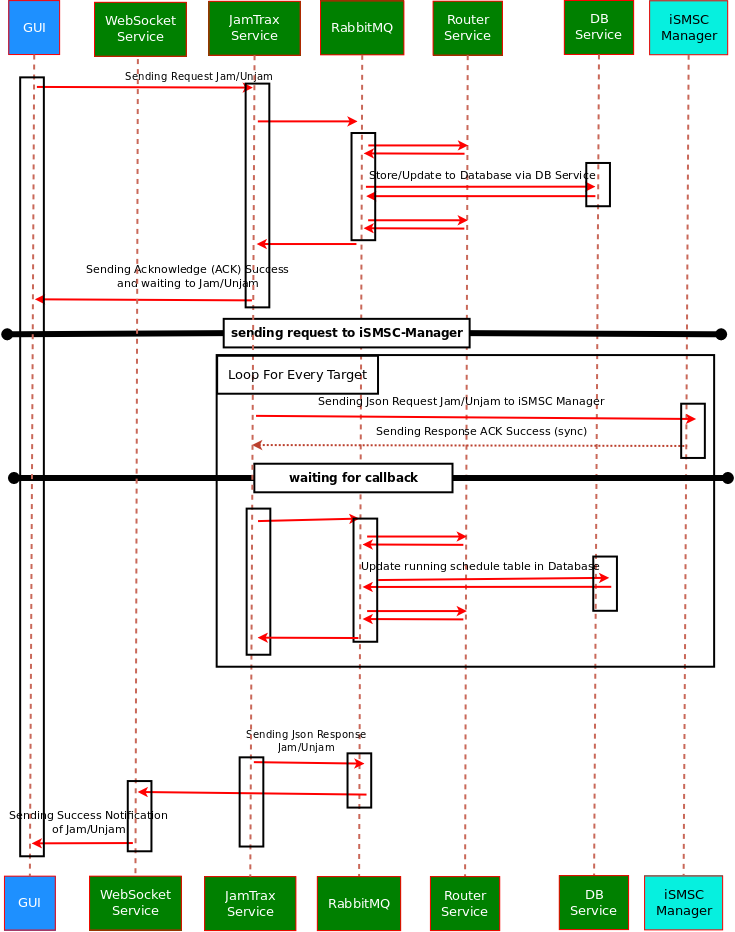
As UI got the success acknowledge from JamTrax Service, this service will start running the scheduler. In this process, JamTrax service will check the schedule that is stored in the database. This checking process is again by passing through Router service.

Then when JamTrax got the value response, JamTrax service will continue running the cron job process by sending the request in json format directly to the iSMSC Manager without using RabbitMQ. This process will need a while to get the callback response from iSMSC Manager.

After JamTrax service got the response, JamTrax will first update the RUNNING\_SCHEDULE table in the database. The related number will be flagged as jammed. Then when the updated is succeed, JamTrax service will continue the response in Json format to the WebSocket service and finally send the response directly to PHP without using RabbitMQ and after that displaying the notification success to UI.



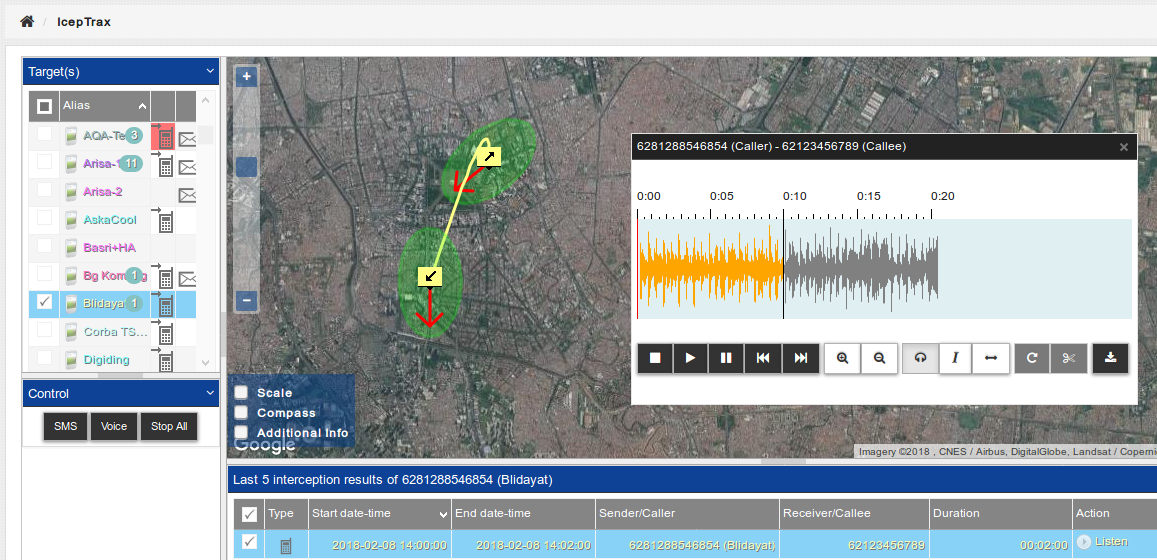
The sequence of Jamming Service for target and ad hoc jamming is divided into 3 parts of process. The first is requested method of jamming process, the second sending request to iSMSC-Manager, and last is the callback. Target and ad hoc jamming do not need to use scheduler and cron job because jamming service will send jam request directly to iSMSC-Manager after receiving inserting target jamming data to database and sending acknowledge notification to UI.



# IcepTrax

**Introduction**

IcepTrax provides interceptions of phone calls by tapping into the operator's network. Telco Dept. takes care of that (with SelVi; Selective Voice Intercept module) and forwards the meta data such as caller location, callee location, and the conversation itself. For streaming the conversation from SelVi to ALTAMIDES we use an [RTP Streaming Server](https://project.1rstwap.com/projects/iceptrax/wiki/RTP_Streaming_Server). In the IcepTrax we provide a graphical user interface to manage the registration of targets and listening the conversations through the browser.

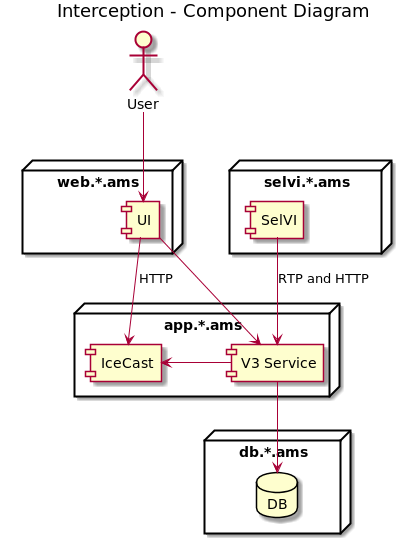


## **SDD**

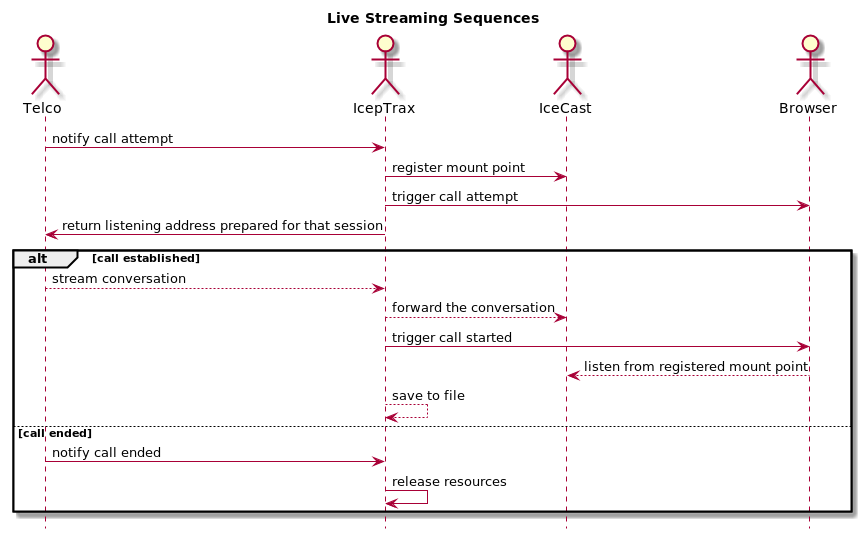
References to the different SDD files in Alfresco related to this module.

* SDD: https://doc-srv.1rstwap.com/share/page/site/development/document-details?nodeRef=workspace://SpacesStore/9d3cc621-f76c-4741-83cc-3ae7fd2fc550

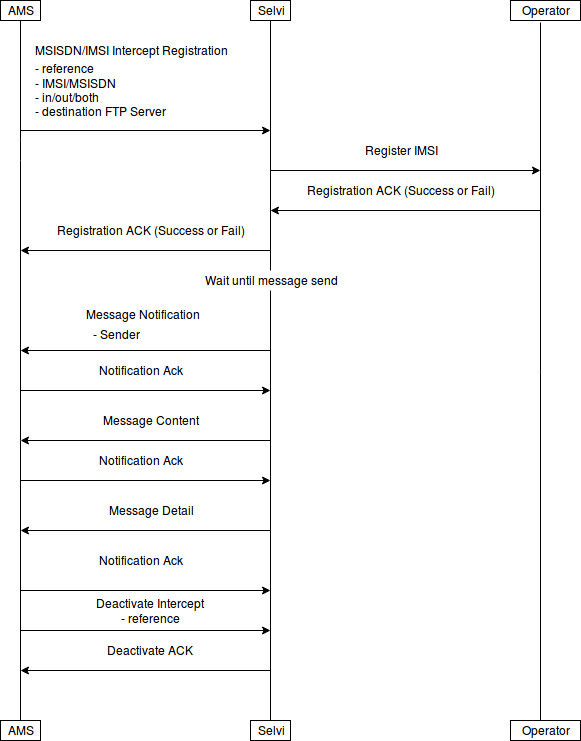
This is how IcepTrax is deployed on ALTAMIDES and how it connects to SelVI:



This chart explains how a live stream is established when SelVi detects an incoming or outgoing phone call:

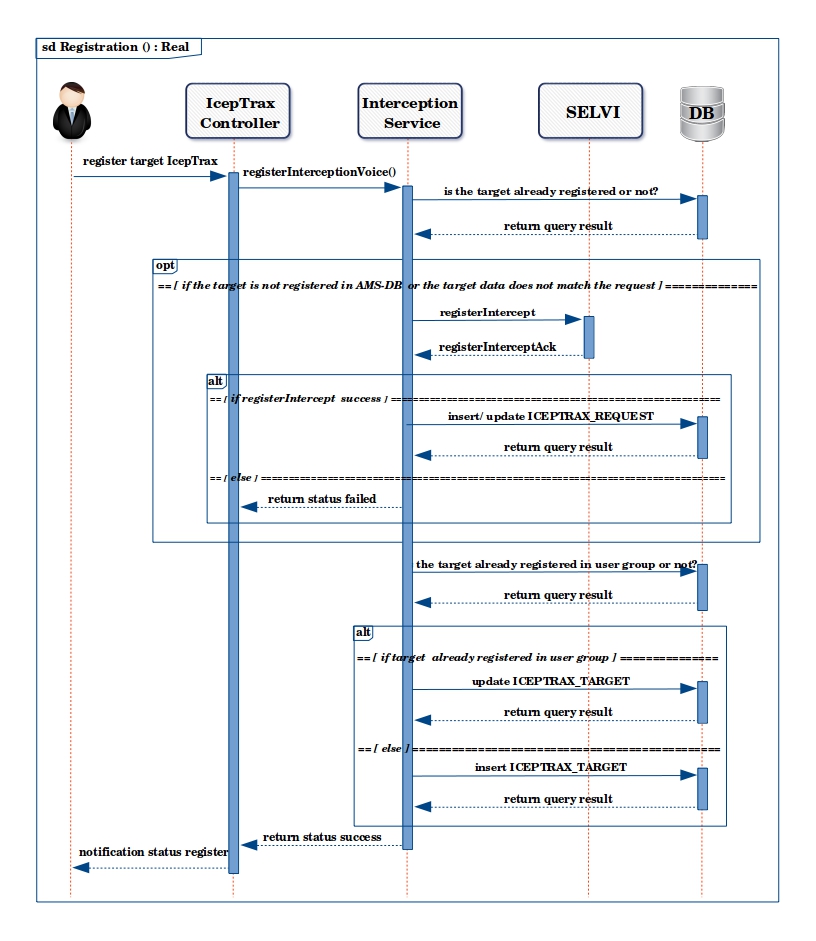


This is how IcepTrax communicates with SelVi for SMS/Data Interception, including registration:



**Registration Flow**

Below is Diagram Sequence for IcepTrax registration:



**Register Request from Front-End**

{

"header": {

"destination": "IceptraxService",

"callback": ",amq.gen-7SHLau3gQ5c0R-A3N48S6g",

"source": "Iceptrax",

"transactionId": "d96f6f95-1829-4b93-8dcb-7243aea19475",

"resolvedDestination": false,

"resolvedCallback": true,

"callbackMsg": false

},

"body": {

"action": "REGISTRATION",

"body": {

"target": ["6281288883001","6281288883002"],

"targetType": ["msisdn","msisdn"],

"interceptVoice": [-1,-1],

"interceptMessage": [3,3],

"metaDataOnly": [false,false],

"dataReroute": [0,0],

"uuid": ["",""],

"vehicleId": ["5617""5619"],

"clientId": 479,

"createdBy": 2029,

"modifiedBy": 2029

}

}

}

**Registration API Request sent to Telco**

{

"version": "1.0",

"uuid": "01d532dd-7762-46ed-b03f-41f483dc67cf",

"type": "registerIntercept",

"async": false,

"callbackUrl": "https://app.alpha.ams:7175/selvi/interception",

"param": {

"target": "601116000000",

"targetType": "msisdn",

"nodes": [

{

"host": "CC"

},

{

"host": "DG"

}

],

"limitations": [

"international"

],

"metaDataOnly": true,

"interception": {

"voice": {

"interceptDirection": "inbound||outbound||both||null"

},

"sms": {

"interceptDirection": "inbound||outbound||both||null"

}

},

"reroute": {

"data": {

"ggsnIp": "10.100.1.10"

}

}

}

}

**RegisterAck**

{

"version": "1.0",

"requestUuid": "01d532dd-7762-46ed-b03f-41f483dc67ab",

"type": "registerInterceptAck",

"status": 0,

"param": {

"interception": {

"voice": {

"interceptDirection": "inbound||outbound||both||null",

"status": 0,

"statusDescription": "Successful Registration"

},

"sms": {

"interceptDirection": "inbound||outbound||both||null",

"status": 0,

"statusDescription": "Successful Registration"

}

}

}

}

registerAck-status:

-1: Invalid request

0: Successful Registration

1: Partially Success

2: Successful to Update Interception

3: Partially Success to Update Interception

4: Failed

5: UUID or MSISDN already registered  
  
registerAck-param.status:

0: Successful Registration

1: Successful to Update Interception

2: Inbound Failed

3: Outbound Failed

4: Both Failed

-1: Not Registered

**Notification to Front-End**

[

transactionId=d96f6f95-1829-4b93-8dcb-7243aea19475,

status=true,

message=null,

body=IcepTraxResponse{

transactionId='null',

status=true,

message='null',

body=[

IcepTraxResponseBody{

uuid='null',

target='6281288883001',

targetType='null',

statusCode=null,

statusDesc='null',

statusReg=false,

statusCodes={

interceptMessage=message\_5,

interceptVoice=message\_5,

dataReroute=message\_9

},

iceptraxResultId=null,

iceptraxRequestId=0,

interceptUuid='null',

interceptType=null,

callingInfo='null',

calledInfo='null',

content='null',

voiceRecorder='<no voice calling data>',

startCallTimestamp='null',

endCallTimestamp='null',

ip='null',

callPort='null',

createdAt='null',

duration='null',

voiceCalled='<no voice called data>',

direction=null,

callRingingTime='null',

callAnsweredTime='null',

callReleasedTime='null',

callReleaseCode='null',

callReleaseDesc='null'

},

IcepTraxResponseBody{

uuid='null',

target='6281288883002',

targetType='msisdn',

statusCode=null,

statusDesc='null',

statusReg=true,

statusCodes={

interceptMessage=message\_5,

interceptVoice=null,

dataReroute=message\_9

},

iceptraxResultId=null,

iceptraxRequestId=0,

interceptUuid='null',

interceptType=null,

callingInfo='null',

calledInfo='null',

content='null',

voiceRecorder='<no voice calling data>',

startCallTimestamp='null',

endCallTimestamp='null',

ip='null',

callPort='null',

createdAt='null',

duration='null',

voiceCalled='<no voice called data>',

direction=null,

callRingingTime='null',

callAnsweredTime='null',

callReleasedTime='null',

callReleaseCode='null',

callReleaseDesc='null'

}

],

allGgsnList=null

}

]

**Intercept Messages:**

inbound\_success = "message\_1";

inbound\_failed = "message\_2";

outbound\_success = "message\_3";

outbound\_failed = "message\_4";

both\_success = "message\_5";

both\_failed = "message\_6";

inbound\_success\_outbound\_failed = "message\_7";

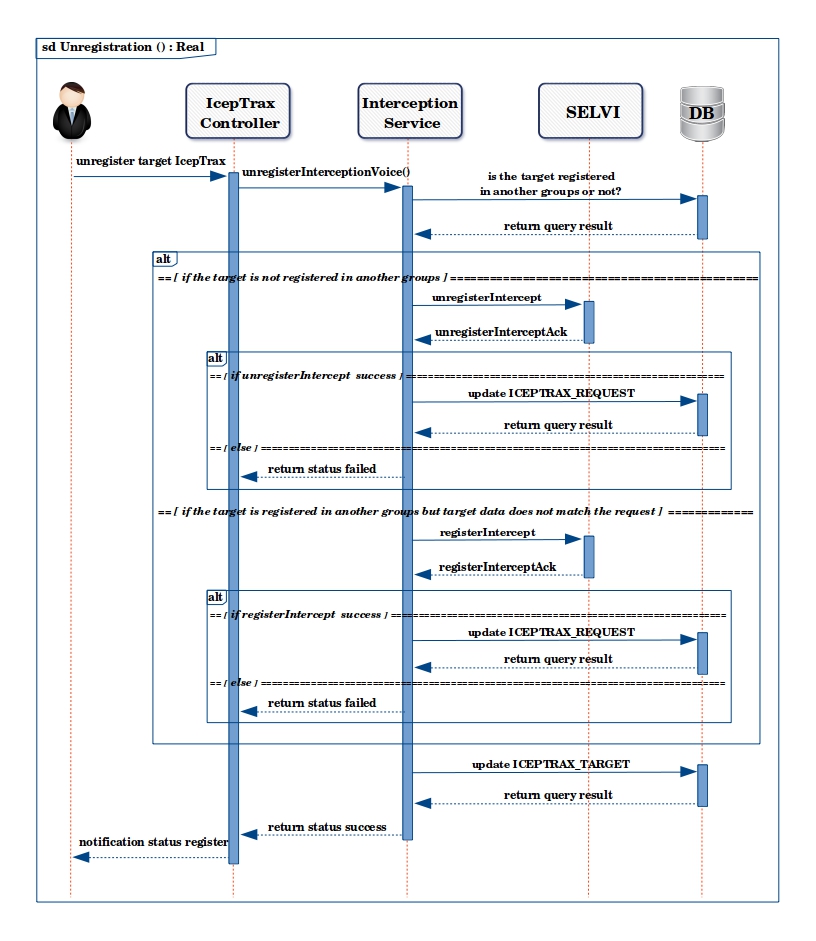
outboud\_success\_inbound\_failed = "message\_8";

reroute\_success = "message\_9";

reroute\_failed = "message\_10";

**Unregistration Flow**

Below is Diagram Sequence for IcepTrax unregistration:



**Unregister Requested**

{

"version": "1.0",

"uuid": "01d532dd-7762-46ed-b03f-41f483dc67cf",

"type": "unregisterIntercept",

"async": false,

"callbackUrl": "https://app.alpha.ams:7175/selvi/interception",

"param": {

"target": "601116000000",

"targetType": "msisdn",

"nodes": [

{

"host": "CC"

},

{

"host": "DG"

}

],

"limitations": [

"international"

]

}

}

**UnregisterACK**

{

"version": "1.0",

"requestUuid": "01d532dd-7762-46ed-b03f-41f483dc67ab",

"type": "unregisterInterceptAck",

"status": 0,

"param": {

"interception": {

"voice": {

"interceptDirection": "inbound||outbound||both||null",

"status": 0,

"statusDescription": "Successful Unregistration"

},

"sms": {

"interceptDirection": "inbound||outbound||both||null",

"status": 0,

"statusDescription": "Successful Unregistration"

}

}

}

}

unregisterAck-status:

-1: Invalid request

0: Successful Unregister Intercept

1: Partially Success Unregister Intercept

2: UUID or MSISDN not registered  
  
registerAck-param.status:

0: Successful Unregistration

1: Successful to Update Interception

2: Inbound Failed

3: Outbound Failed

4: Both Failed

-1: Not Registered

**Interception Steps**

**Voice**

1. Call Attempt
2. Stream Start
3. Call Details
4. Call Ended

**SMS**

1. Message Attempt
2. Message Content
3. Message Details

**Port Range**

The port range in which we define to SelVi the ports needed to stream the caller and callee(s) is 10000 – 12000.

**Audio File Storage**

When an RTP stream occurs, the gstreamer plugin installed on the app server is configured to do the following three steps:

1. Transcode the stream to an OGG (Vorbis) temporary audio file (filesink) in the /app/tmp/ams-service-interception/audio folder. The /app/temp/ams-service-interception/audio folder is mounted as a different partition in the app server. By default it should have at least 2GB of storage allocation, that means it can store up to one thousand 60-minute phone calls per 5 minutes.
2. Forward the stream to IceCast for live listening on the GUI (shout2send)
3. Send to IceCast to listen to the stream (appsink)

After the phone call has ended, the temporary audio file is copied into the database as a blob. Five minutes after the phone all is ended, the temporary audio file is automatically removed from the /app/tmp/ams-service-interception/audio folder.

We use gStreamer to decode the audio over RTP packets. We have tested so far the follow audio types:

1. MULAW
2. ALAW
3. AMR-NB (Narrow Band)
4. AMR (Wide Band)

To find a full list of supported types (and plugins), please see:  
<https://gstreamer.freedesktop.org/documentation/plugins.html>

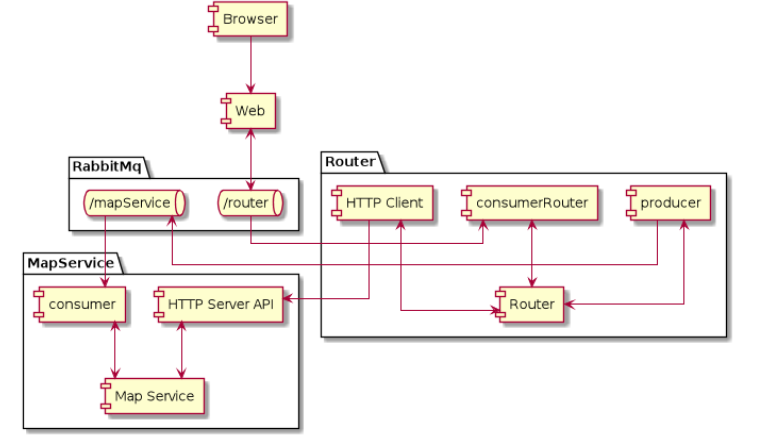
Document related to IcepTrax:

* <https://project.1rstwap.com/projects/iceptrax/wiki/IcepTrax>
* https://doc-srv.1rstwap.com/share/page/site/development/documentlibrary#filter=path%7C%2FALTAMIDES%2FIcepTrax%7C&page=1

# Map Services

Map services is one of ATLAMIDES module that currently has draw cell and draw arrow capabilities. The Draw Cell capability will provide cell coordinate with specific shape. Map service will defines which qualities are available for certain cell reference, and calculate all latitude and longitude for drawing cell coverage. The Draw Arrow capability provides track path between one latitude longitude to another latitude longitude. It will give latitude longitude with certain density and arrow.

The following diagram illustrates the component relationship for Map Visualization:



## **SDD**

References to the different SDD files in Alfresco related to this module.

* SDD: https://doc-srv.1rstwap.com/share/page/site/development/document-details?nodeRef=workspace://SpacesStore/3709117d-6cf4-4429-9484-26d689b16b5d

**LQV**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lat/long type | Radius | Azimuth | End angle | Type |
| BTS | > 0 | 0-359 | 0-359 | CCC (directional) |
| BTS | > 0 | -1 | NULL (not used for omnidirectional antennas) | CCC (omnidirectional) |
| BTS | NULL (will use default 300m) | 0-359 | 0-359 | ECC |
| BTS | NULL (will use default 300m) | 0-359 | NULL (will use by default: azimuth + 60 degrees) | ECC |
| BTS | > 0 | 0-359 | NULL (will use by default: azimuth + 60 degrees) | ECC |
| BTS | > 0 | NULL (will use by default: end angle - 60 degrees) | 0-359 | ECC |
| BTS | NULL (will use default 300m) | -1 | NULL (or ignored if filled) | ECC (omnidirectional |
| Estimated | > 0 | NULL | NULL | ECA |
| BTS | NULL | NULL | NULL | BTS |
| BTS | > 0 | NULL | NULL | BTS |

* We do not allow a radius of 0 meters!
* In all above cases a latitude/longitude values must exist.

### Directional CCC (Calculated Cell Centre)

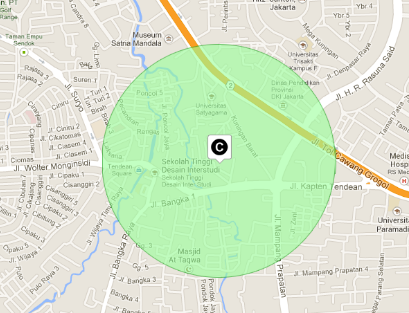
CCC (Calculated Cell Centre) location quality is used when information about the cell to accurately calculate its shape and centre coordinates are known enough. To do so, we need to know: the location of the BTS tower, the direction of the antenna (azimuth), the beam width and power output (radius) of the antenna.



* BTS Latitude/Longitude
* Have Radius
* Have azimuth (0 - 359)
* Have end angle, otherwise default end angle of (azimuth + 60 degrees is used)

### Omni directional CCC (Calculated Cell Centre)

The Omnidirectional CCC (Calculated Cell Centre) location quality is used in cases where cell antenna is omnidirectional. It means the antenna does not have a specific direction (azimuth), and the beam width is 360 degrees. Such cell should be represented by a circle instead of a sector. Since cell centre of such cell has the same coordinates as the BTS tower, it doesn't make sense to show BTS location marker.



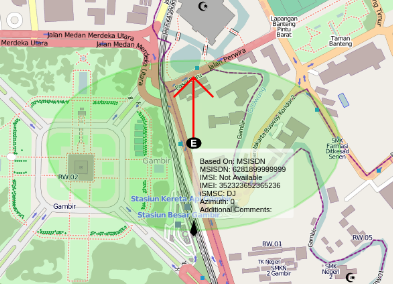
* BTS Latitude/Longitude
* Have Radius
* Azimuth is -1 (means 360 degrees)

### ECC (Estimated Cell Centre)

ECC (Estimated Cell Centre) location quality is used when the location of the BTS tower and the direction of the antenna (azimuth) are known, but lacking of other information, such as the power output (radius) of the antenna, beam width, tilt, etc.

In order to display more than just BTS tower location, we assume the values with the most commonly used values, which is 120-degree beam width and 300-metre radius.. This, however, is just an estimation, thus the centre of such cell is called Estimated Cell Centre (ECC).

The ECC is displayed as a sector with its radius originating from the BTS tower location and the Azimuth “red” arrow points always from the cell center into the azimuth direction. We also mark the BTS tower location with a special icon.



* BTS Latitude/Longitude
* No Radius (NULL) will use default radius = 300m
* Have Azimuth 0-359

### BTS (BTS Location)

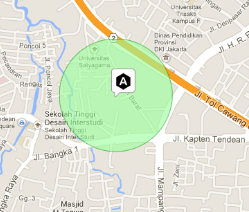
BTS is a piece of equipment that facilitates wireless communication between user equipment (UE) and a network. In our case, however, we refer to the BTS tower holding the antenna.

The BTS location quality is used when only BTS tower coordinates of the retrieved CGI are known. That means we know the location of the tower, but not the direction (azimuth) or power (radius) of the antenna.

* BTS Latitude/Longitude
* No Azimuth (NULL)
* No Radius (NULL)

### ECA (Estimated Cell Area)

When cell coverage information is gathered from third parties or by means of 'wardriving'/'netmonitoring'/'cellspotting', the BTS tower location is usually not known. In such cases we display a circle approximating the cell coverage area and call it Estimated Cell Area (ECA). With ECA the only information we have is the approximate centre and radius of the cell coverage area.

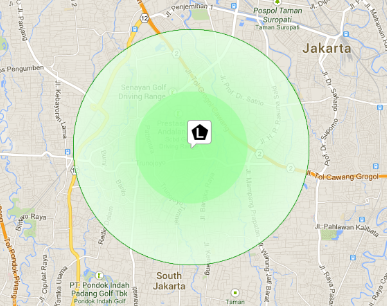


* Cell Latitude/Longitude
* Have Radius (Radius > 0)
* Azimuth is unknown (NULL), but for the visualization we show a circle (360 degrees)

### LA (Location Area)

LA is a logical grouping of cells in close geographical proximity to each other. Each LA within operator's network has a unique code, called Location Area Code (LAC). This code is the third element in the Cell Global Identification (CGI): 510.10.1045.31562. All cells that belong to the same LA will also have the same LAC in their cell reference (a.k.a. CGI).

If during the location query we retrieve the CGI, but do not know the exact coordinates of the cell, we can estimate the centre and coverage area of the LA based on the known coordinates of other cells in the same LA. We can also estimate the likely radius of the LA coverage area from other Las. Thus LA can be shown on a map as two concentric circles: a smaller one, covering all known cells in this LA, and a larger one, with radius equal to the average or maximum size of LAs in the network.



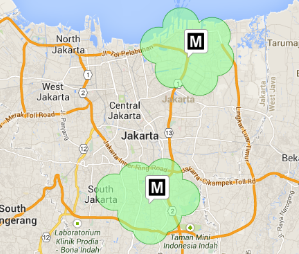
* No Cell Ref data
* Location Area data is available

### MSC (Location Area)

MSC is the second lowest location quality of target location after 'Country'. It means that we were able to determine the MSC which is currently serving the target, but the geographical area covered by the MSC can be very large, potentially as big as the country itself.

MSC is responsible for a number of location areas (LA). When a location query is successful, the link between MSC managing the target and LA the target resides in is noted and stored. Later this relationship information is used to estimate the coverage shape of an MSC by combining the shapes of the related LAs.

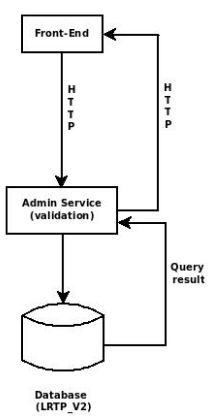
Sometimes, due to high load for example, one MSC can also be responsible for LAs of another MSC. In such cases, it is possible that the MSC coverage shape will be divided into several large LA clusters. These LA clusters can also be quite far apart from each other.



* No Cell Ref data
* No Location Area Data
* MSC data is available

# Admin Service

For the Admin module, an Admin Service will be implemented. The communication between the Front-End and the Admin Service will be through the HTTP channel. If we use admin service for ADMIN module, the validation will be processed using the service itself. So we do not have to validate using JavaScript. In other words, we will change to back-end for input validation. Please see the diagram below.



## **SDD**

References to the different SDD files in Alfresco related to this module.

* SDD: https://doc-srv.1rstwap.com/share/page/site/development/document-details?nodeRef=workspace://SpacesStore/89cd2e3c-a9e6-4e04-9ed5-1d94effcc661

**Admin Service API**

For Admin Service API, it requires secured HTTP protocol (https). It is important to note that the security configuration is centralized. In the sense that the file .keystore must be stored in the folder /app/servers/configuration. The location is defined in application.properties for each micro-service configuration.

For attributes that have the DateTime data type it is required that the Admin service converts the value to GMT timezone prior to saving it into the database. That’s why all DateTime references in to API use the ISO 8601 standard, meaning the timezone is always included. For example: 2018-12-17T04:30:53+07:00.

**Devices List**

REQUEST URI:

https://app.client.ams:9002/device/list

REQUEST METHOD:

POST

REQUEST HEADER:

Content-Type: application/json

SAMPLE REQUEST (listDevices) for Group Admin

{

"deleted": false,

“clientId”: 1

}

SAMPLE RESPONSE (listDevices)

[

{

"body": {

"status": true,

"result": [

{

"deviceId": 46,

"deviceName": "iPhone XXXtes",

"description": "iPhone XXX",

"imei": "012753002111186",

"startTime": "2019-01-09T00:00:00Z",

"endTime": "2020-01-07T14:00:00Z",

"isActive": true,

"deleted": false,

"createdBy": 2029,

"createdByName": "superadmin",

"createdAt": "2019-01-07T12:32:20Z",

"clientId": 4,

"clientName": "Demo Test",

"modifiedBy": 2029,

"modifiedByName": "superadmin",

"modifiedAt": "2019-10-10T07:23:26Z",

"types": [

{

"deviceTypeId": 1,

"deviceTypeName": "MobileTrax"

}

]

},

{

"deviceId": 51,

"deviceName": "esia hidayah",

"description": "esia hidayah",

"imei": "01275300211118333",

"startTime": "2019-11-19T10:00:00Z",

"endTime": "2019-12-19T10:00:00Z",

"isActive": false,

"deleted": false,

"createdBy": 2029,

"createdByName": "superadmin",

"createdAt": "2019-01-08T10:35:21Z",

"clientId": 4,

"clientName": "Demo Test",

"modifiedBy": 2029,

"modifiedByName": "superadmin",

"modifiedAt": "2019-07-30T10:32:01Z",

"types": [

{

"deviceTypeId": 1,

"deviceTypeName": "MobileTrax"

}

]

}

]

}

}

]

**Groups List**

REQUEST URI:

https://app.client.ams:9002/group/list

REQUEST METHOD:

POST

REQUEST HEADER:

Content-Type: application/json

SAMPLE REQUEST (groupsList)

{

“deleted”: false

}

SAMPLE RESPONSE (groupsList)

{

"status": true,

"result": [

{

"clientId": 1,

"clientName": "Admin Group"

},

{

"clientId": 2,

"clientName": "Superadmin Group"

}

]

}

**View Device Details**

REQUEST URI:

https://app.client.ams:9002/device/get

REQUEST METHOD:

POST

REQUEST HEADER:

Content-Type: application/json

SAMPLE REQUEST (getDevice)

{

"deviceId": 1

}

SAMPLE RESPONSE (getDevice)

{

"status": true,

"deviceLicenseLimit": 10,

"deviceTotalGlobal": 8,

"deviceTotalGroup": 6,

"result": [

{

"deviceId": 1,

"deviceName": "iPhone X",

"deviceDescription": "iPhone X",

"imei": "3589160502645301",

"startTime": "2018-11-21T10:00:00+00:00",

"endTime": "2018-12-21T10:00:00+00:00",

"isActive": true,

"deleted": false,

"client": {

"clientId": 1,

"clientName": "MOI-EGYPT"

},

"created": {

"userId": 1,

"userName": "satrio",

"date": "2018-12-21T10:00:00+00:00"

},

"modified": {

"userId": null,

"userName": null,

"date": null

},

"deviceTypes": [

{

"deviceTypeId": 1,

"deviceTypeName": "MobileTrax",

"assignedToUserInGroup": true

},

{

"deviceTypeId": 2,

"deviceTypeName": "SecomTrax",

"assignedToUserInGroup": true

},

{

"deviceTypeId": 3,

"deviceTypeName": "CellTrax",

"assignedToUserInGroup": false

},

{

"deviceTypeId": 4,

"deviceTypeName": "SecomTrax",

"assignedToUserInGroup": false

}

]

}

]

}

**Save New Device**

REQUEST URI:

https://app.client.ams:9002/device/save

REQUEST METHOD:

POST

REQUEST HEADER:

Content-Type: application/json

SAMPLE REQUEST (saveDevice)

{

"imei": "3589160502645301",

"deviceName": "iPhone 9",

"description": "iPhone 9",

"startTime": "2018-11-19T10:00:00",

"endTime": "2018-12-19 10:00:00",

"isActive": "1",

"createdBy": 141,

"createdAt": "2018-11-19 10:00:00",

"clientId": 1,

"deviceTypes": [1,2]

}

SAMPLE RESPONSE (saveDevice)

{

"status": true,

"message": ”notif\_device\_added”,

“items”: {

“device”: “iPhone9”

}

}

**Update Device**

REQUEST URI:

https://app.client.ams:9002/device/update

REQUEST METHOD:

POST

REQUEST HEADER:

Content-Type: application/json

SAMPLE REQUEST (updateDevice)

{

"deviceId": 1,

"imei": "3589160502645301",

"deviceName": "iPhone 9",

"description": "iPhone 9",

"startTime": "2018-11-19T10:00:00+07:00",

"endTime": "2018-11-19T10:00:00+07:00",

"isActive": "1",

"modifiedBy": "157",

"modifiedAt": "2018-09-10T07:14:49+00:00",

"clientId": 1,

"deviceTypes": [

1,

2

]

}

SAMPLE RESPONSE (updateDevice)

{

"status": true,

"message": ”notif\_device\_modified”,

“items”: {

“device”: “iPhone 9”

}

}

**Activate Device**

REQUEST URI:

https://app.client.ams:9002/device/activate

REQUEST METHOD:

POST

REQUEST HEADER:

Content-Type: application/json

SAMPLE REQUEST (activateDevice)

{

"body": {

"deviceId": [1,2,3]

“clientId”: 1

}

}

SAMPLE RESPONSE (activateDevice)

{

"result": [{

"status": true,

"message": "notif\_device\_activated",

"items": {

"device": "iPhone 9"

}

},

{

"status": true,

"message": "notif\_device\_activated",

"items": {

"device": "iPhone X"

}

},

{

"status": false,

"message": "err\_device\_activation",

"items": {

"device": "Samsung Galaxy series"

}

}]

}

**Delete Device**

REQUEST URI:

https://app.client.ams:9002/device/delete

REQUEST METHOD:

POST

REQUEST HEADER:

Content-Type: application/json

SAMPLE REQUEST (deleteDevice)

{

"body": {

"deviceId": [1,2,3]

“clientId”: 1

}

}

SAMPLE RESPONSE (deleteDevice)

{

"result": [{

"status": true,

"message": "notif\_device\_deleted",

"items": {

"device": "iPhone 9"

}

},

{

"status": true,

"message": "notif\_device\_deleted",

"items": {

"device": "iPhone X"

}

},

{

"status": true,

"message": "notif\_device\_deleted",

"items": {

"device": "Samsung Galaxy series"

}

}]

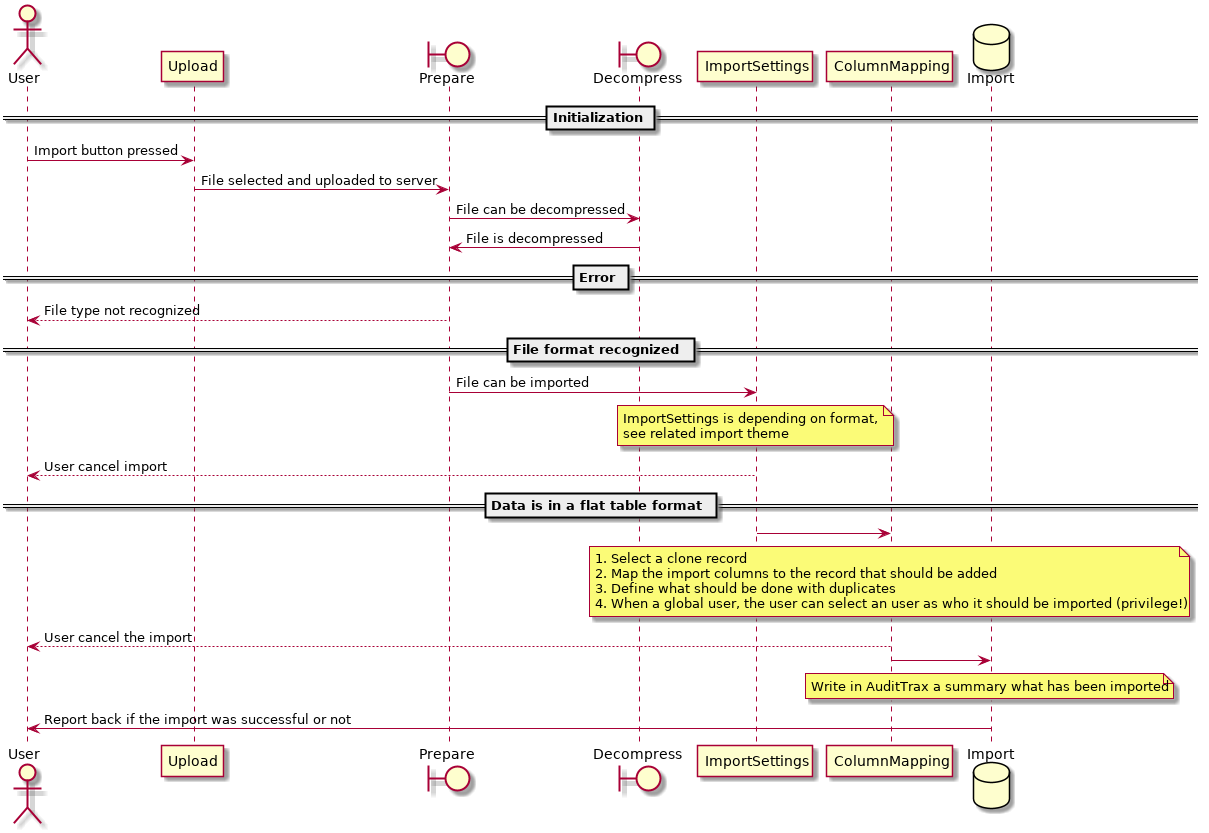
}

# Import Wizard

# **Introduce**

Import Wizard is the tool for import data on ProfileTrax – Basic Target Admin which are using jqWidgets, Lumen microframework and Dropzone that makes user can import import data more dynamically.

The sequence diagram shows on a higher level the steps that will happen:



## **Described flow**

### Start import

The user clicks the import button or select import in the grid menu. On this action the import wizard will be started.

### Select one file

The import wizard screen is opened.

The user can upload one file by clicking the drop zone and select a file, or by drag and drop the file in the drop zone.

This component also has a progress bar, that jqWidgets does not support.

[< Back] is disabled  
[Next >] is enabled when one file was successfully selected / uploaded.  
[Cancel] is always enabled (on select the uploaded file (when available) should be deleted / archived)

With next you go to "3. data preparation".

### Data preparation

The data preparation process happens on the server. The file was successful uploaded and the user selected next. The server will analyze the uploaded file and uses an internal factory to automatic detect the uploaded information. The process will automatically decompress compressed files. Expected is that only one file is provided in a compressed file. If not, the wizard should jump back to "2. Select one file" and feedback to the user the file cannot be imported.

When the uploaded data is not uniquely detected by the factory what kind of format it is, go to step "4. File type selection".

When the uploaded data is successfully detected by the factory which importer should be used, jump to "5. Import wizard" when that imported needs some settings, otherwise jump to "6. Clone".

### File type selection

This is a user interaction in the wizard. The server could not detect what kind of file was uploaded. The user sees a list of possibilities, provided by the server, what kind of import filters are available (and possible applying). The user must select one option to be capable to select next.

[< Back] deletes / archives the current file and jump to "2. Select one file".  
[Next >] is enabled when one import filter is selected and jumps to "5. Import wizard" when the import filter needs some settings, otherwise jump to "6. Clone".  
[Cancel] is always enabled (on select the uploaded file should be deleted / archived)

### Import wizard

The import wizard is an iframe managed by the server to define how the uploaded data for the import should be interpreted. This is depending on the import filter. Such as the CSV import filter. In the future we should be capable to support multiple import formats.

The next and back should run through the steps of the import filter wizard, but should also be capable to jump back to a previous page. The next should in the end run into "6. Clone".

### Clone

To make import easier with incomplete information, we introduce the clone functionality. With this clone the user can select one of the records from the original list to use as a template. Like the clone button with the grid itself.

In this menu, front-end will request action “getAllColumnValue” to back-end to list all available target within user’s scope for cloning reference.

It is possible to NOT select a clone record (default).

The back button should work as the flow should go. The next button will bring us in "7. Column mapping".

The cancel button will close the wizard and archive / delete the import file.

### Column mapping

At this stage the wizard will map the columns of the system to the columns of the file that needs to be imported. To do this, front-end will request action “getColumn” to back-end to get all valid column. The screen will show a grid with in the first column the column name as used in the system (not necessary the column name used in the database!). The second column is a drop-down list with source selection. The source selection can consist out of the following options:

1. None
2. Filename
3. Clone record name (only when a clone record was selected)
4. Manual input

The third column change based on the source selection.

|  |  |
| --- | --- |
| Source | Import |
| None | Column is empty |
| Filename | Select the column, based on column naming of the importer |
| Clone | Column has fixed text with the same column name |
| Manual input | Free text input, where the user can type a text |

The fourth column is used for formatting the import. At the moment the column has a different type as the input column (e.g. date / text). Here we can select how the input should be interpreted.

The fifth till seventh columns will show examples of the import based on the first three rows of the import file.

The default selection is based on smart matching. Keep in mind that sometimes we need a virtual column for importing. Such as for the Cell DB. This to select on which level the Cells should be stored for selection.

In this menu we can also validate first 3 data by pressing [Validate] button. This will call action “previewData” from back-end to validate the data.

next will bring the wizard into "8. Duplicates"   
back and cancel will work as all the previous times.

### Duplicates

This screen is used to define what the import function should do when a duplicate record is found between the import and the database. Even when one record is found double in the import file. Duplicate is defined by the system.

The user must select one of the options and press next.

### Data validation

This screen is used to define what should be done when input values are wrongly validated.

### Import as

This screen is optional when the user has a global access on the current import and group / user levels exists, import wizard will do action “getUser” to get all available user within logged in user’s scope. This way the user can choose with which user they want to import the csv.

#### Import overview

At this stage the user will get a summary on what is about to be imported. The next button is now a finish button. On pressing finish the real importing will start in background. A notification will be opened showing the import progress.

The import happens in the background and the user will be notified by the Progress Notification as long the user did not close it. When not closed and the progress is done, the server will close this notification.

When ready a notification is sent that the data has been imported with the summary of the results. Or an error when data was rolled back or partly imported (validation error).

The summary says:  
filename that was imported

1. of records processed
2. records added
3. records updated
4. records skipped
5. records failed in validation  
   Total importing time

### AKKA Actor

Akka is an open-source library that helps to easily develop concurrent and distributed applications using Java or Scala by leveraging the Actor Model. Import wizard use akka to utilize multithreading ability in java to optimize importing speed by using all available cpu cores. Import wizard has 5 actor class for multithreading.

ImportManagerActor. This is parent class for import actor. This class will dispatch new actor for importing process.

ImportActor. This class will do initial process of import, upload and validating the file, and then will dispatch new import task actor for each row in file.

ImportTaskActor. This class will do the importing process. Each row will be handled by one instance of this class.

ImportAliasManagerActor. This is parent class for actors that will handle alias.

ImportAliasActor. This actor will sort gather rows with same alias in to the same thread, this is to prevent duplicate alias.

# **What If Import Wizard Did Not Work**

1. Wrong value in AMS\_CONFIG.CONFIG\_KEY
   * Check CONFIG\_KEY for example
   * CONFIG\_KEY\_NAME = "altamides.use.importwizard", MODULE's value should be "ADMIN" not "GLOBAL".
2. Permission denied to create Lumen session, log and cache
   * Change permission to be chmod -R 777 to folder /www/altamides/app/web/importwizard/storage/
3. Permission denied to save temporary import file
   * Change permission to be chmod 777 to folder /www/altamides/app/web/importwizard/tmp
4. Permission denied to create jqwidgets symlink in Lumen
   * Change permission to be chmod 777 to folder /www/altamides/app/web/importwizard/public/js
5. Database AMS\_MASTER and/or AMS\_CONFIG is not up to date
6. Missing vendor for Lumen
   * You can see <https://lumen.laravel.com/docs/5.1> for how to install Lumen, Install it in folder /www/altamides/app/web/importwizard
7. Config value for group\_concat\_max\_len in MariaDB is too small
   * By default, value for group\_concat\_max\_len in MariaDB is 1024, it is to small for Import Wizard, we need to update it to be 1 000 in file my.cnf
8. Import wizard fail to validate file
   * Please check importwizard config in application.properties and refer to this config: import.service.tmp.file.download.
   * Make sure value/directory that written there is exist in the app server , in this case: import.service.tmp.file.download=//app//tmp//ams-service-importwizard//
   * If it does not exist, please create those folders under /app  
     cd /app && mkdir tmp  
     cd /app/tmp && mkdir ams-service-importwizard
9. Invalid routing key or hostname on /www/altamides/app/web/importwizard/config/config\_import.php
   * make sure hostname and routing key already directed to the correct one.