TigaseDoc

Release 0.1

Tigase, Inc.

Jun 23, 2023

CONTENTS

CHAPTER

ONE

OVERVIEW

Tigase Unified Archive Component is extended version of Tigase Message Archiving Component allow you to store not only messages but also information about:

- login and logout events
- jingle call events
- offline messages
- metadata of transferred file

It is based on extended implementation of XEP-0136: Message Archiving with few custom features allowing more advance searches.

CHAPTER

TWO

ANNOUNCEMENT

2.1 Major changes

Tigase Unified Archive component has undergone a few major changes to our code and structure. To continue to use Tigase Unified Archive component, a few changes may be needed to be made to your systems. Please see them below:

2.1.1 Database schema changes

We decided to no longer use *in-code* database upgrade to update database schema of Unified Archive component and rather provide separate schema files for every supported database.

Additionally we moved from *in-code* generated SQL statements to stored procedures which are now part of provided database schema files.

To continue usage of new versions of Unified Archive component it is required to manually load new component database schema, see *Preparation of database* section for informations about that.

Warning: Loading of new database schema is required to use new version of Unified Archive component.

2.2 New features

2.2.1 Support for using separate database for different domains

Since this version it is possible to use separate archived messages based on domains. This allows you to configure component to store archived message for particular domain to different database.

For more informations please look into [using separate store for archived messages]

CHAPTER

THREE

FEATURES

Tigase Unified Archive Component contains few additional features useful for working with archives containing not only messages but also other informations.

3.1 Archive of Jingle call events

For instructions on how to enable Jingle call archive see *Enabling Jingle call archive* section.

3.1.1 Retrieving a list of collections

To retrieve collections with informations about Jingle call events `item-type` attribute needs to be added to `<list/>` element from section 7.1 Retrieving a List of Collections of XEP-0136 specification with value set to `call`.

A result it will be returned as specified in XEP-0136, but `<list/>` element will contain `<events/>` element for each collections instead of `<chat/>` element. Additionally `<list/>` element will have `item-type` attribute set to `call`.

Example

Request to retrieve collection of calls with `juliet@capulet.com` between `2014-01-01 00:00:00` and `2014-05-01 00:00:00`

```
<iq type="get" id="query1">
    <list xmlns='urn:xmpp:archive'
      with='juliet@capulet.com'
      start='2014-01-01T00:00:00Z'
      end='2014-05-01T00:00:00Z'
      item-type='call'/>
</iq>
```

Response with single collection starting at `2014-02-01 13:01:34`

```
<iq type="result" id="query2">
    <list xmlns='urn:xmpp:archive' item-type='call'>
        <events with='juliet@capulet.com' start='2014-02-01T13:01:34Z'/>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index='0'>0</first>
            <last>0</last>
            <count>1</count>
        </set>
```

(continues on next page)

(continued from previous page)

```
    </events>
</iq>
```

3.1.2 Retrieve a list of entries from collection

To retrieve items with informations about Jingle call from collection `item-type` attribute needs to be added to `<retrieve/>` element from section 7.2 Retrieving a Collection of XEP-0136 specification with value set to `call`.

A result will be returned as specified in XEP-0136, but instead of `<chat/>` element `<events/>` element will be returned. Moreover instead of message subelements in `<from/>` and `<to/>` elements, there will be custom element containing information about call, ie. Jingle session ID - `<sid/>`.

Example

Request to retrieve calls events with `juliet@capulet.com` between `2014-02-01 13:00:00` and `2014-05-01 00:00:00`

```
<iq type="get" id="query1">
    <retrieve xmlns='urn:xmpp:archive'
      with='juliet@capulet.com'
      start='2014-02-01T13:00:00Z'
      end='2014-05-01T00:00:00Z'
      item-type='call'/>
</iq>
```

Response with single entry of incoming call at 2014-02-01 13:01:34 with call SID.

```
<iq type="result" id="query2">
    <events xmlns='urn:xmpp:archive' with='juliet@capulet.com' start='2014-02-
↪01T13:00:00Z' item-type='call'>
        <from secs='94' item-type='call'><sid>SID_OF_CALL</sid></from>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index='0'>0</first>
            <last>0</last>
            <count>1</count>
        </set>
    </events>
</iq>
```

3.2 Archive of login/logout events

For instructions on how to enable login history please see *Enabling login history* section.

3.2.1 Retrieving a list of collections

To retrieve collections with informations about user logins `item-type` attribute needs to be added to <list/> element from section 7.1 Retrieving a List of Collections of XEP-0136 specification with value set to `login`.

A result it will be returned as specified in XEP-0136, but <list/> element will contain <events/> element for each collections instead of <chat/> element. Additionally <list/> element will have `item-type` attribute set to `login`.

Example

Request to retrieve collection of login events of `juliet@capulet.com` between `2014-01-01 00:00:00` and `2014-05-01 00:00:00` (sent from `juliet@capilet.com` account)

```
<iq type="get" id="query1">
    <list xmlns='urn:xmpp:archive'
      with='juliet@capulet.com'
      start='2014-01-01T00:00:00Z'
      end='2014-05-01T00:00:00Z'
      item-type='login'/>
</iq>
```

Response with single collection starting at `2014-02-01 13:01:34`

```
<iq type="result" id="query2">
    <list xmlns='urn:xmpp:archive' item-type='login'>
        <events with='juliet@capulet.com' start='2014-02-01T13:01:34Z'/>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index='0'>0</first>
            <last>0</last>
            <count>1</count>
        </set>
    </events>
</iq>
```

3.2.2 Retrieve a list of entries from collection

To retrieve items with informations about user logins from collection `item-type` attribute needs to be added to <retrieve/> element from section 7.2 Retrieving a Collection of XEP-0136 specification with value set to `login`.

A result will be returned as specified in XEP-0136, but instead of <chat/> element <events/> element will be returned. Moreover instead of message subelements in <from/> element, there will be custom element containing information about session, ie. <bind/> element with connection resource set as a value. There will be also `type` attribute added to <from/> element with value set to `login` to mark that this is `login` event.

Example

Request to retrieve login events of `juliet@capulet.com` between `2014-02-01 13:00:00` and `2014-05-01 00:00:00` (sent from `juliet@capilet.com` account)

```
<iq type="get" id="query1">
    <retrieve xmlns='urn:xmpp:archive'
      with='juliet@capulet.com'
      start='2014-02-01T13:00:00Z'
      end='2014-05-01T00:00:00Z'
      item-type='login'/>
</iq>
```

Response with single login entry at 2014-02-01 13:01:34 with call SID.

```
<iq type="result" id="query2">
    <events xmlns='urn:xmpp:archive' with='juliet@capulet.com' start='2014-02-
↪01T13:00:00Z' item-type='login'>
        <from secs='94' item-type='login'><bind>balcony</bind></from>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index='0'>0</first>
            <last>0</last>
            <count>1</count>
        </set>
    </events>
</iq>
```

3.3 Retrieval of all events and messages

It is also possible to retrieve full history of login/logout events, and history of Jingle calls and messages.

3.3.1 Retrieve a list of entries

To retrieve items with informations about every archived entry `item-type` attribute needs to be added to `<retrieve/>` element from section 7.2 Retrieving a Collection of XEP-0136 specification with value set to `any`.

A result will be returned as specified in XEP-0136, but instead of `<chat/>` element `<events/>` element will be returned. `<from/>` and `<to/>` elements will have `item-type` attribute set to `message`, `login`, `logout` or `call` depending on type of returned event. Subelements of `<from/>` and `<to/>` elements will depend on type of archived event it represents.

Example

Request to retrieve events of `juliet@capulet.com` between `2014-02-01 13:00:00` and `2014-05-01 00:00:00` (sent from `juliet@capilet.com` account)

```
<iq type="get" id="query1">
    <retrieve xmlns='urn:xmpp:archive'
      with='juliet@capulet.com'
      start='2014-02-01T13:00:00Z'
```

(continues on next page)

(continued from previous page)

```
      end='2014-05-01T00:00:00Z'
      item-type='any'/>
</iq>
```

Response with single login entry at 2014-02-01 13:01:34 with call SID.

```
<iq type="result" id="query2">
    <events xmlns='urn:xmpp:archive' with='juliet@capulet.com' start='2014-02-
↪01T13:00:00Z'>
        <from secs='34' item-type='message'><body>Example message</body></from>
        <from secs='64' item-type='call'><sid>SID_OF_CALL</sid></from>
        <from secs='94' item-type='login'><bind>balcony</bind></from>
        <set xmlns="http://jabber.org/protocol/rsm">
            <first index='0'>0</first>
            <last>2</last>
            <count>3</count>
        </set>
    </events>
</iq>
```

3.4 Retrieval of recent events

It is now possible to retrieve list of last logged events (chat, calls, groupchats) for every bare jid with which archive owner communicated with. As this collection may be rather big, we added support for XEP-0059: Result Set Management to make it possible to request part of a list (ie. first 100 entries). For now it is only possible to request next entries of a list by providing offset/index at which next result should start at.

> **Warning:** There is no support for RSM pagination done using <before/> and <after/>.

3.4.1 Retrieval of list of all recent events

In this example we will request and receive recent events of any type or condition from whole time for which we have entries in an archive.

Example request to retrieve full list of all recent events.

```
<iq type="get" id="query1">
    <query xmlns="urn:xmpp:tigase:recent"/>
</iq>
```

Example response to retrieval of all recent events.

```
<iq id="1" xmlns="jabber:client" type="result">
    <events xmlns="urn:xmpp:tigase:recent">
        <event
              with="buddy-2@example.com"
              type="call"
              direction="from"
```

(continues on next page)

(continued from previous page)

```
                condition="success"
                time="2017-01-21T10:06:10Z" />
        <event
                with="some-room@muc.example.com"
                type="groupchat"
                direction="from"
                body="Welcome to our groupchat"
                time="2017-01-20T10:08:12Z"/>
        <event
                with="buddy-1@example.com"
                type="chat"
                direction="to"
                body="Hello world"
                time="2017-01-19T17:20:51Z" />
    </events>
</iq>
```

3.4.2 Retrieval of list of all recent events in specified time period

In this example we will request and receive recent events of any type or condition which occurred within specified time frame.

Example request to retrieve full list of all recent events.

```
<iq type="get" id="query1">
    <query xmlns="urn:xmpp:tigase:recent" start="2017-01-20T00:00:00Z" end="2017-01-
↪31T00:00:00Z"/>
</iq>
```

Example response to retrieval of all recent events.

```
<iq id="1" xmlns="jabber:client" type="result">
    <events xmlns="urn:xmpp:tigase:recent">
        <event
                with="buddy-2@example.com"
                type="call"
                direction="from"
                condition="success"
                time="2017-01-21T10:06:10Z" />
        <event
                with="some-room@muc.example.com"
                type="groupchat"
                direction="from"
                body="Welcome to our groupchat"
                time="2017-01-20T10:08:12Z"/>
    </events>
</iq>
```

3.4.3 Requesting filtered list of recent events

It is also possible to filter list of recent events by event type

- chat
- groupchat
- call (and call status)
  - success (successful call)
  - missed (missed call)
  - canceled (canceled call)

Requesting only recent chat events

Example request to retrieve full list of all recent chat events.

```
<iq type="get" id="query1">
    <query xmlns="urn:xmpp:tigase:recent">
        <event-types>
            <chat/>
        </event-types>
    </query>
</iq>
```

Example response to retrieval of recent chat events.

```
<iq id="1" xmlns="jabber:client" type="result">
    <events xmlns="urn:xmpp:tigase:recent">
        <event
               with="buddy-2@example.com"
               type="chat"
               direction="from"
               condition="success"
               time="2017-01-21T09:06:10Z" />
        <event
               with="buddy-1@example.com"
               type="chat"
               direction="to"
               body="Hello world"
               time="2017-01-19T17:20:51Z" />
    </events>
</iq>
```

Requesting only recent missed or canceled call events

Example request to retrieve list fo recent missed or canceled call events.

```
<iq type="get" id="query1">
    <query xmlns="urn:xmpp:tigase:recent">
        <event-types>
            <call>
                <missed/>
                <canceled/>
            </call>
        </event-types>
    </query>
</iq>
```

Example response to retrieval of recent missed or canceled call events.

```
<iq id="1" xmlns="jabber:client" type="result">
    <events xmlns="urn:xmpp:tigase:recent">
        <event
               with="buddy-2@example.com"
               type="call"
               direction="from"
               condition="missed"
               time="2017-01-20T10:06:10Z" />
        <event
               with="buddy-1@example.com"
               type="call"
               direction="to"
               condition="canceled"
               time="2017-01-18T17:20:51Z" />
    </events>
</iq>
```

3.5 Retrieval of transferred files metadata

It is also possible to query metadata of archived messages containing details of a transferred files. This metadata may be retrieved/queried by:

- archive owner **(by default)**
- admin owner **(if ``allow-admin-querying-users-data`` is set to ``true``)**

As returned result set may be quite large, mechanism supports XEP-0059: Result Set Management for pagination.

Note: All timestamps sent to the server or returned by the server using this extension are compatible with DataTime profile defined in XMPP Data and Time Profiles (XEP-0082)

Returned result set may be filtered with use of the following fields.

Name	Field type	Expected value	Description	Access
domain	list-single	Domain for which request sender is an admin	Return only files sent to/from a domain	Admin
owner	jid-single	User from a domain for which request sender is an admin	Return only files sent to/from a user	Admin
after	text-single	Timestamp	Returns only files sent after this timestamp	User / Admin
before	text-single	Timestamp	Returns only files sent before this timestamp	User / Admin
contains	text-single	Text	Returns only files for which name, description or url contain this text	User / Admin
media-type	text-single	Text	Returns only files for which media type contain this text	User / Admin
smaller-than	text-single	Number	Returns only files for which size is smaller than	User / Admin
bigger-than	text-single	Number	Returns only files for which size is bigger than	User / Admin

Note: If field will not be set, then its value will not be used to filter results.

Tip: If `owner` field is set, then `domain` field may be left empty.

Warning: If admin of a domain will not pass `domain` or `owner` it will receive results only from his own archive (files sent to or by him).

3.5.1 Retrieval of form fields

To know which fields you may specify to query archived files metadata, send following query:

Example request to retrieve form fields.

```
<iq type="get" id="queryForm1">
    <query xmlns="urn:xmpp:tigase:file:metadata"/>
</iq>
```

The server replies with all the form fields it support in queries:

Example response on form fields retrieval.

```
<iq type="result" id="queryForm1">
    <query xmlns="urn:xmpp:tigase:file:metadata">
        <x xmlns="jabber:x:data" type="form">
            <field var="FORM_TYPE" type="hidden">
                <value>urn:xmpp:tigase:file:metadata</value>
            </field>
            <field var="domain" type="list-single" label="Domain">
```

(continues on next page)

(continued from previous page)

```
                <option>example.com</option>
            </field>
            <field var="owner" type="jid-single" label="Owner"/>
            <field var="after" type="text-single" label="After timestamp"/>
            <field var="before" type="text-single" label="Before timestamp"/>
            <field var="contains" type="text-single" label="Text to search for in name,␣
↪description or url"/>
            <field var="media-type" type="text-single" label="Part of media type␣
↪(MimeType)"/>
            <field var="smaller-than" type="text-single" label="Maximal size of a file"/>
            <field var="bigger-then" type="text-single" label="Minimal size of a file"/>
        </x>
    </query>
</iq>
```

Note: Fields `domain` and `owner` will available only when query would be executed by a user which is a domain administrator of any vhost at the installation.

3.5.2 Retrieval of transferred files metadata

To do that we send `<iq/>` of type `set` with `<query/>` element qualified with by namespace `urn:xmpp:tigase:file:metadata` which must contain data form with `FORM_TYPE` field set to `urn:xmpp:tigase:file:metadata`. The form may contains additional fields, but we may skip those fields for which we do not wish to provide values.

Optionally, to paginate or limit number of results, we may add `<set/>` qualified by namespace `http://jabber.org/protocol/rsm` as specified in XEP-0059: Result Set Management.

Example request to retrieve first page of results..

```
<iq type="set" id="query1">
    <query xmlns="urn:xmpp:tigase:file:metadata">
        <x xmlns="jabber:x:data" type="submit">
            <field var="FORM_TYPE" type="hidden">
                <value>urn:xmpp:tigase:file:metadata</value>
            </field>
        </x>
        <set xmlns='http://jabber.org/protocol/rsm'>
            <max>10</max>
        </set>
    </query>
</iq>
```

In the response server will return a `<list/>` element qualified by namespace `urn:xmpp:tigase:file:metadata`. In this element there will be a `<set/>` element qualified by `http://jabber.org/protocol/rsm` (see XEP-0059: Result Set Management) containing pagination details and a list of `<item/>` elements. Each `<item/>` element will contain following attributes: - `id` - ID of the message containing metadata - `with` - bare jid of a sender/recipient of a file - `stamp` - timestamp of a message - `url` - link to download the file

Additionally, domain admin will receive `owner` attribute which will contain bare jid of the owner of the archive in which result was found.

Each <item/> element will contain file element qualified by namespace urn:xmpp:jingle:apps:file-transfer:5 which may contain following elements:

- media-type
- name
- desc
- size

The <file/> element format is described in XEP-0234: Jingle File Transfer - 5. Application Format.

Warning: <file/> element will contains <name/>, <desc/>, <size/> and <media-type/> elements only if file transfer was initiated with XMPP <message/> containing description of a transferred file specified in XEP-0385: SIMS

Example response with first page of results.

```
<iq type="set" id="query1">
    <list xmlns="urn:xmpp:tigase:file:metadata">
        <item id="dcbb06cb-1d41-41bf-a4f3-396af26d0509"
            with="buddy-1@example.com"
            stamp="2021-04-28T08:15:04.063Z"
            url="https://example.com/uploaded/b2718a48-ed88-4afe-99c2-ee99219d896f/first.
↪jpg">
            <file xmlns="urn:xmpp:jingle:apps:file-transfer:5">
                <media-type>image/jpeg</media-type>
                <name>first.jpg</name>
                <desc>First shared image</desc>
                <size>1334232</size>
            </file>
        </item>
        <item id="a8c131e8-5ed5-46dd-aef7-871fd88531ab"
            with="buddy-2@example.com"
            stamp="2021-04-28T09:25:04.563Z"
            url="https://example.com/uploaded/c5d0bc87-b3ef-4846-aee9-2855e067f03a/
↪second.jpg">
            <file xmlns="urn:xmpp:jingle:apps:file-transfer:5"/>
        </item>
        ...
        <item id="f4805382-da50-46e5-a683-70dbd214a0cb"
            with="buddy-1@example.com"
            stamp="2021-04-28T12:53:14.862Z"
            url="https://example.com/uploaded/19d07312-6c10-40c0-b185-735fed7d452e/
↪flower.jpg">
            <file xmlns="urn:xmpp:jingle:apps:file-transfer:5">
                <media-type>image/jpeg</media-type>
                <name>flower.jpg</name>
                <desc>Image of a flowers</desc>
                <size>1274422</size>
            </file>
        </item>
        <set xmlns='http://jabber.org/protocol/rsm'>
            <first index="0">dcbb06cb-1d41-41bf-a4f3-396af26d0509</first>
```

(continues on next page)

(continued from previous page)

```
            <last>f4805382-da50-46e5-a683-70dbd214a0cb</last>
            <count>123</count>
        </set>
    </query>
</iq>
```

Tip: To retrieve next page of the results, send request with the same query but add <after/> or <before/> elements with proper values to the set element of the request.

CHAPTER

FOUR

DATABASE

4.1 Preparation of database

Before you will be able to use Tigase Unified Archive Component and store messages and events in particular database you need to initialize this database. We provide few schemas for this component for MySQL, PostgreSQL, SQLServer and DerbyDB.

They are placed in `database/` directory of installation package and named in `dbtype-unified-archive-version.sql`, where `dbname` in name of database type which this schema supports and `version` is version of a Unified Archive Component for which this schema is designed.

You need to manually select schema for correct database and component and load this schema to database. For more information about loading database schema look into *[Databse Preparation]* section of *preparation of database*

4.2 Upgrade of database schema

Database schema for our components may change between versions and if so it needs to be updated before new version may be started. To upgrade schema please follow instuctions from *Preparation of database* section.

Note: If you use SNAPSHOT builds then schema may change for same version as this are versions we are still working on.

4.3 Schema description

Tigase Unified Archive component uses few tables and stored procedures and it shares same tables with Tigase Message Archiving component - in fact it uses same tables with addition of few fields. Due to that used tables are prefixed with `tig_ma_` and `tig_ua_`, but used stored procedures are prefixed with `Tig_UA_`.

4.3.1 Table `tig_ma_jids`

This table stores all jids related to stored messages and events, ie. from `to` and `from` attributes of archived stanzas.

Field	Description	Comments
jid_id	Database ID of a JID	
jid	Value of a bare JID	
jid_sha1	SHA1 value of lowercased bare JID	Used for proper bare JID comparison during lookup. (Not exists in PostgreSQL schema)
domain	Domain part of a bare JID	Stored for easier lookup of messages owned by users of a particular domain

4.3.2 Table `tig_ma_msgs`

Table stores archived events.

Field	Description	Comments
stable_id	ID of a stored event	Unique with matching `owner_id`
owner_id	ID of a bare JID of a event owner	References `jid_id` from `tig_ma_jids`
buddy_id	ID of a bare JID of a event recipient/sender (different than owner)	References `jid_id` from `tig_ma_jids`
ts	Timestamp of a event	Timestamp of archivization or delayed delivery
body	Body of a message	
msg	Serialized event	
stanza_id	ID attribute of archived event	
is_ref	Marks if message is a reference to other message	
ref_stable_id	`stable_id` of referenced message	
itemtype	Event type	May be one of: `0` - message of unknown type, `1` - chat message, `2` - groupchat message, `132` - call, `128` – `login`, `` `129 `` - logout. (Added in UA schema)
offline	Marks offline events	0 - not offline 1 - offline and will be stored after delivery 2 - offline and will be removed after delivery

4.3.3 Table `tig_ma_tags`

Table stores tags of archived messages. It stores one tag for many messages using `tig_ma_msgs_tags` to store relation between tag and a message.

Field	Description	Comments
tag_id	Database ID of a tag	
owner_id	ID of a bare JID of a tag owner	ID of bare JID of owner for which messages with this tag were archived
tag	Actual tag value	

4.3.4 Table `tig_ma_msgs_tags`

Table stores relations between tags and archived messages with this tags.

Field	Description	Comments
msg_owner_id	ID of a bare JID of a tag owner	ID of bare JID of owner for which messages with this tag were archived
msg_stable_id	ID of a stored event	Unique with matching `msg_owner_id`
tag_id	Database ID of a tag	References `tag_id` from `tig_ma_tags`

4.3.5 Table `tig_ua_for`

Table stores resource of a sender of a message saved to the offline storage.

Field	Description	Comments
stable_id	ID of a stored event	Unique with matching `owner_id`
owner_id	ID of a bare JID of a event owner	References `jid_id` from `tig_ma_jids`
buddy_res	Resource part of a event recipient/sender JID	

4.3.6 Table `tig_ua_jingle`

Table stores jingle metadata from a message saved to the storage.

Field	Description	Comments
stable_id	ID of a stored event	Unique with matching `owner_id`
owner_id	ID of a bare JID of a event owner	References `jid_id` from `tig_ma_jids`
direction	Direction of event	0 - sent by owner 1 - recevied by owner
offline	Marks call to offline	1 if was to offline user
sid	SID of a Jingle call	
action	Jingle action	
reason	Reason of action	Usually for termination/decline events

4.3.7 Table `tig_ua_file_metadata`

Table stores jingle metadata from a message saved to the storage.

Field	Description	Comments
stable_id	ID of a stored event	Unique with matching `owner_id`
owner_id	ID of a bare JID of a event owner	References `jid_id` from `tig_ma_jids`
url_hash	SHA1 of the URL	
url	URL to download sent file	
name	File name	If available
description	Description of a file	If available
media_type	Type of a file (MIMETYPE)	If available
size	File size	If available

CHAPTER

FIVE

CONFIGURATION

To enable Tigase Unified Archive Component you need to add following block to `etc/init.properties` file:

```
unified-archive () {
}
```

It will enable component and configure it under name `unified-archive`. By default it will also use database configured as `default` data source to store data.

Due to fact that Tigase Unified Archive component is extended version of Tigase Message Archiving component it shares configuration properties.

Every configuration property of `message-archive` component may be set to `unified-archive` component. In same way every configuration property of `message-archive` processor may be set to `unified-archive` processor.

5.1 Support for XEP-0136

To be able to use Unified Archive component with XEP-0136: Message Archiving protocol, you additionally need to enable `unified-message-archive-xep-0136` SessionManager processor:

```
sess-man {
    unified-message-archive-xep-0136 () {
    }
}
```

5.2 Enabling Jingle call archive

To enable archiving of Jingle calls you need to enable `jingle-archive` SessionManager processor:

```
sess-man {
    jingle-archive () {
    }
}
```

Additionally you may want to replace `urn:xmpp:jingle:1` XMLNS in entries stored in Unified Archive. For that you need to set `omitJingleXMLNS` of `jingle-archive` processor to `true`.

If you want to archive also Jingle requests incoming by `<message/>` stanzas you need to set `archiveJingleMessage` of `jingle-archive` processor to `true`.

5.2.1 Example

In this example we are enabling processor and all it's options.

```
sess-man {
    jingle-archive () {
        omitJingleXMLNS = true
        archiveJingleMessage = true
    }
}
```

5.3 Enabling login history

To enable archiving of login/logout events you need to enable `login-history` SessionManager processor:

```
sess-man {
    login-history () {
    }
}
```

CHAPTER

SIX

LIMITATIONS

- Component groups messages in collections using date of a messages instead of id of message thread, due to fact that some clients are sending messages with no thread id (ie. Psi, Psi+).
- Only bare JID is stored of sender or recipient.