

Tarîqa 3.0, Installation Manual

Copyright

© EUROPEAN COMMISSION, 2007-2010

© EUROPEAN EXTERNAL ACTION SERVICE, 2011-2012

Licensed under the EUPL, Version 1.1 only (the “Licence”).

You may not use this work except in compliance with the Licence.

You may obtain a copy of the Licence at: <http://ec.europa.eu/idabc/eupl>

Unless required by applicable law or agreed to in writing, software distributed under the Licence is distributed on an "AS IS" basis, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the Licence for the specific language governing permissions and limitations under the Licence.

Authors of Tarîqa:

Pascal Havelange, Pascal.HAVELANGE@eeas.europa.eu

Cosmin Popescu, Adrian.POPESCU@ext.eeas.europa.eu

Table of Contents

I. Introduction.....	6
I.1. Description.....	6
I.2. Features.....	6
I.2.a, A rich source of information.....	6
I.2.b, High quality.....	6
I.2.c, Useable.....	6
I.2.d, Customizable.....	7
I.2.e, Easy to use.....	7
I.2.f, Cost-effective.....	7
I.2.g, Secure and reliable.....	8
I.2.h, Public Administration Reference.....	8
I.3. Path.....	8
I.4. Architecture Overview.....	8
I.4.a, The Tariqa website (the public site).....	8
I.4.b, The Tariqa SOAP Webservice (the web-service).....	9
I.4.c, The Tariqa Service (the private site).....	9
I.5. Engines, sources, queries,	10
I.5.a, Engines.....	10
I.5.b, Sources.....	10
I.5.c, Articles.....	10
I.5.d, Queries & Unique Query Syntax.....	11
I.5.e, Filters.....	11
I.5.f, Standard Pages.....	11
I.5.g, Special Pages.....	11
I.5.h, Flavours.....	12
I.6. Speed improvement through mDdownloader and tariqa3.0_service.....	12
I.6.a, The model.....	12
I.6.b, AJAX, mDownloader.....	12
Without Ajax:.....	12
With Ajax “only”:.....	13
With Ajax and mDownloader:.....	13

I.6.c, SOAP Protocol, application authentication amongst the Webservice.....	14
I.6.d, Caching.....	14
I.7. Authorization & Access levels.....	14
I.7.a, Four Access levels.....	15
I.7.b, And a fifth one.....	16
II. Prerequisites.....	17
III. Installation.....	18
III.1. Steps.....	18
III.2. Application's configuration.....	19
III.3. Webservice's configuration.....	20
III.3.a, Tariqa Webservice.....	20
III.3.b, TariqaAuthentication Webservice.....	21
IV. Customization.....	22
IV.1. Custom authentication mechanism.....	22
IV.1.a, Default mechanisms.....	22
ECAS/LDAP.....	22
Internal Database.....	22
IV.1.b, Custom mechanism.....	23
iAuthentication/tariqaAuthentication.....	23
iUser/tariqaUser.....	23
IV.2. Content sharing & LDAP.....	23
IV.3. Adding custom pages to Tariqa: the “special pages”.....	24
IV.3.a, Special page creation, step by step.....	24
IV.3.b, Inherited Methods.....	26
getNews().....	26
getDocumentRepository().....	26
IV.4. Adding an Engine to the system.....	27
IV.5. Adding an Export Format to the system.....	28

I. Introduction

I.1. Description

Tarîqa is an open source intelligence platform developed by the Crisis Room of the European Commission's Directorate-General for External Relations. It is a heuristic tool “ supported by a multimedia content database“ that facilitates search, investigation, analysis and discovery. The ultimate aim of the platform is to provide real-time support for early warning and crisis response.

I.2. Features

I.2.a, A rich source of information

- Access to full-text databases such as Lexis Nexis, Oxford Analytica, Factiva, Google,...
- Integration RSS feeds, Wikipedia, ...
- It is possible to integrate any source of information which provides an access protocol or an API; through some coding, it is therefore even possible to integrate any internal data as a source for Tarîqa (multimedia content, satellite imagery, internal news feeds or publications, ...).
- Users can upgrade the keywords that filter their preferred content at any time.

I.2.b, High quality

Although **Tarîqa** "casts its net widely", the information it pulls up is of a high quality:

- Only the sources you have selected are used.
- Analysis-orientated and authoritative
- Focused on testimonials, documentaries, academic research and investigative journalism.

I.2.c, Useable

Tarîqa's advanced information retrieval tools make it possible to discern useful knowledge from the masses of information that are available.

- End-user may use the Queries [pre-]defined by the system manager, or may create their own at any time.
- Information from Queries is automatically filtered and ranked in terms of relevance.

I.2.d, Customizable

- **Tarîqa's** functions are crafted to meet the needs of both State and non-State actors engaged in risk assessment, crisis response, peace-building, and scenario planning.
- Pages can be easily maintained and updated as the geopolitical situation evolves.

I.2.e, Easy to use

- **Tarîqa's** only requires familiarity with using a web browser.
- Provides multiple views on the information: quick overviews, in-depth analysis, project and policy orientated content.
- “Flat structure”. All information on a particular topic is accessible from a single page and with just one click of your mouse.
- Core content is quickly accessible, even through a slow internet connection

I.2.f, Cost-effective

- All you need to begin using **Tarîqa's** is a web browser.
- **Tarîqa** is a web application; there is no need to install or maintain software on individual client computers*.
- Because the web-based user interface is highly intuitive, users can begin to use the system after only minimal training, and they can quickly develop the skills, methodologies and geopolitical culture to extract more value from the system.

*Users who wish to use the suggested "full cycle workflow" would need specific softwares such as a Mind Map editor or a statistical suite (EIDOS).

I.2.g, Secure and reliable

- **Tarîqa** can be used as open source software. It can be bundled or unbundled and used on any organization's networks.
- Systems Security depends from the network infrastructure used to host the service. The European external Action Service deploys **Tarîqa** through an encrypted SSL connection.

I.2.h, Public Administration Reference

The **Tarîqa** platform is developed by the EUROPEAN EXTERNAL ACTION SERVICE.

Before the creation of the EEAS, the project was developed by the: EUROPEAN COMMISSION - EXTERNAL RELATIONS DIRECTORATE GENERAL - Crisis Platform and Policy Coordination in CFSP Crisis Response and Peace Building.

I.3. Path

In the current document please consider the following *placeholders* as:

- *InstallationRoot* the path where you have downloaded **Tarîqa**.
- *ServerRoot* the path where you are installing **Tarîqa**.
- *DocumentRoot* the configured *DocumentRoot*¹ of your Apache HTTP server.
- *mDownloaderRoot* the path where *mDownloader* is installed
- *mAggregatorRoot* the path where *mAggregator* is installed
- *TariqaUrl* the URL used to access your instance of **Tarîqa**.

I.4. Architecture Overview

The **Tarîqa** web application is actually composed of (but not limited to) three main “components”:

I.4.a, The Tarîqa website (the public site)

This is the User Interface & Access point of the application. As such, this is the only component that will be accessible to your user community.

¹ <http://httpd.apache.org/docs/2.2/mod/core.html#documentroot>

It is a website written with the `php 5` programming language, and which uses `Javascript` to improve the user's browsing experience.

The code is based on a small `php` framework developed by the **Tarîqa** programmers². Thanks to the improved `object oriented model of php 5`, the whole application is written as a hierarchy of `classes`, which should be easy to understand and eventually to modify by any `php` developer.

The website is “internationalizable”, as all textual content of the site is stored in a database. If more than one language versions were encoded in the database, the end-user would even be allowed to select by her-/himself the language of the Tarîqa website. Please note that, the language of the interface is not related in any way to the language of the content displayed inside the interface (by example, nothing impedes you to translate the interface to Kiswahili³, while your all Engines (see I.5.a) return information in English only. In such a case, the end-user will be able to browse Tarîqa in its favorite language, but the article displayed, being available only in English, will be displayed in the English language.

I.4.b, The Tarîqa SOAP Webservice (the web-service)

This is the “engine”, powering the Tarîqa website. The Webservice acts as an interface between the Tarîqa Website and the Engines.

In fact, the Tarîqa Website does not know anything about the various protocols & mechanisms involved in communicating with the various Engines. Instead, it knows the API provided by the Webservice. This API tend to deliver the services of each engine to the Tarîqa Website in a uniform way. Whatever the proprietary data format/protocol of the Engines, the data is first processed by the Webservice and returned to the Tarîqa Website in well defined XML format.

In addition of isolating the display (the website) and the processing (the webservice), this method offers the possibility to develop a fully customized user-interface or even a brand new software (be it a new website, or a desktop application), that would connect and use the Engines of Tarîqa,

I.4.c, The Tarîqa Service (the private site)

This is an additional layer between the Tarîqa Website and the Tarîqa Webservice; its only purpose

2 Pascal Havelange: Pascal.HAVELANGE@eeas.europa.eu;

Cosmin A. Popescu: Adrian.POPESCU@ext.eeas.europa.eu

3 Muzuri Kabisa! (“Very Good”!)

is to speed up internal Queries processing on the main server by, either bypassing the SOAP protocol (calling directly the local functions); either bypassing the Webservice authentication mechanism (calling specific versions of the SOAP methods).

More information in the paragraph I.6.c.

I.5. Engines, sources, queries, ...

The engines, sources and queries are the way through which the **Tarîqa** application will fetch the news articles or data from its various sources.

I.5.a, Engines

What we call an “Engine” is the entity that provides the information to **Tarîqa**. As such it is in general a company (like “Lexis Nexis”, “Google”, “Oxford Analytica”, ...), which offers a mean to interrogate their content through a specific protocol.

Each “Engine” is capable of providing content from at least one “Source”. The number of “Sources” in an engine is not limited by **Tarîqa**.

I.5.b, Sources

A “Source”, in the vocabulary of **Tarîqa**, is a publication (much like a newspaper or a press agency), that provide content (i.e.: articles), and which is offered by an “Engine”. By example, it could be “The BBC Monitoring” (which, in our case, is provided by the Lexis Nexis “Engine”); but it could also be “all the articles written in the French language” provided by the Google News “Engine”.

I.5.c, Articles

An “Article” is a unit of content (in general, the text of a press article), as provided by a specific “Source”.

Please note that **Tarîqa** does not require “Articles” to be made of text. Any data received from a “Source” is considered an “Article”. By example, a piece of video, is an “Article” from multimedia “Source”.

I.5.d, Queries & Unique Query Syntax

A “Query” is a set of words that can be understood by an “Engine” in order to fetch relevant “Articles” from a specific “Source” or from a set of “Sources”.

The syntax of a “Query” is usually specific to each “Engine”. Nevertheless, **Tarîqa** includes a specific algorithm that allows users to communicate with all engines through a single/common syntax. This mechanism is called the “UQS” (**Unified Query Syntax**), and it basically allows **Tarîqa** to convert Queries between the Engine's specific syntax and the UQS.

I.5.e, Filters

A “Filter” in its function, is very close to a “Query”, with the notable exception that “Filters” are defined by the End-users themselves, and can therefore be applied on top of existing “Queries”.

This mechanism allows the End-users to customize their browsing experience on **Tarîqa**.

I.5.f, Standard Pages

Tarîqa offers two type of “Standard Pages” to the End-users:

- Country Pages
- Thematic Pages

Both have similar function: offer to the End-user, access through one single page, to all the relevant information on a specific subject of interest: a geographical location (i.e. Angola) or a specific thematic (i.e. Energy Security).

To perform this task, the team in charge of maintaining **Tarîqa** defines and maintains, inside the:

```
tariqapro database
```

a set of “Queries”, which have been elaborated and fine-tuned to match the needs of most End-users.

I.5.g, Special Pages

As a simple way of extending **Tarîqa**, the system contains templates of so-called “Custom Pages” that can be fully rewritten by a web developer, and integrated into the application.

Pages written using that mechanism can benefit, with no efforts, from most functions of **Tarîqa**

(Access to sources, information sharing mechanism, document repository, authentication/authorization, ...).

By example, Tariqa contains a “Custom Page”, available at:

`TARIQAURL/special_page.php?module=weboffice&`

I.5.h, Flavours

A “Flavour” is a way of categorizing information on the “Country Pages”, by grouping the predefined “Queries” by subject.

By example, most “Country Pages” provide the following “Flavours”: “General News”, “Economic News”, “Violence”, ... “Category”.

I.6. Speed improvement through mDdownloader and tariqa3.0_service

As speed is key factor for the user's browsing experience, we have implemented advanced mechanisms inside **Tariqa** in order to keep loading time as low as possible.

I.6.a, The model

Three major implementation decisions have been taken in order to (greatly) improve the overall performance of the system:

I.6.b, AJAX, mDownloader⁴

As some “Engines” (i.e.: Lexis Nexis) perform much slower than others, and sometime even spend minutes before returning any result; we have adopted an asynchronous model; powered by Javascript AJAX on the client side, and a custom Windows Service (mDownloader) on the server side.

Lets imagine that a specific “Country Page” is composed of ten “Queries”. The result of each of them will be used to fill-in the page:

Without Ajax:

the client web browser would process the queries sequentially (one after the other, waiting for the full results to be received before moving to the next query), and eventually display the final

⁴ Mdownloader is also available under the EUPL license. For more information, please consult:

<https://joinup.ec.europa.eu/software/mdownloader/description>

page after all Queries have been processed.

This is unacceptable as loading time of the page would be far too long.

Even worst, the whole page would fail to load if any of the ten “Queries” would fail (because of a temporary network failure; because a specific engine is in maintenance state; ...).

With Ajax “only”:

in principle the 10 Queries could be performed in parallel; which would at least solve the speed issue.

Unfortunately by design, most modern browser limit the number of concurrent HTTP requests sent simultaneously to one specific server to two requests.

Therefore, the loading time would actually be reduced but never less than 50% of the original loading time (as at most two requests will be performed simultaneously).

Knowing that some Engines perform very slow, in some condition, the total loading time could even be “as slow” as the sequential way (it happens, by example, if, out of the 10 queries, the first 2 to be processed are the 2 slowest).

With Ajax and mDownloader:

in order to cope with this problem, we have developed an independent `Windows Service`: `mDownloader`, its role is to perform the HTTP requests instead of the end-user's browser. This service is fully configurable. It can be set to manage as many concurrent connection as possible with the Server's available bandwidth. The result of each request is then made available to **Tarîqa**. Using this mechanism, the client's browser will still issue ten HTTP requests, through `Ajax`; but each request will do not more than instructing `mDownloader` to perform a Query. In no occasion will it need to wait for the results.

As a result the page loading time is not anymore affected by the number or the type of queries to be displayed.

On the client side, As soon as the container page's structure is completely loaded by the user's browse, some `JavaScript` code will query the `mDownloader` service in a loop to get any available result.

As a result, the page is loaded as fast as if no Queries where involved; the Queries' result are displayed as soon as they are made available by `mDownloader` in a fully asynchronous way (without affecting each other).

I.6.c, SOAP Protocol, application authentication amongst the Webservice

As pages are often composed of tenth of “Queries” and each queries are served internally as a distinct HTTP request to the **Tarîqa** Webservice, using the SOAP protocol. The fact of using the SOAP protocol implies a small overhead to each and every function call as XML/SOAP packets must be successively generated and parsed by the client application and the Webservice host.

In addition, at every function call, the Webservice checks the authentication and authorization of the client application.

Although these actions only represents a fractions of a second; once multiplied by the number of requests per page, its has a non negligible impact on the performances.

In a setup where the **Tarîqa** Website is hosted on the same server as the **Tarîqa** Webservice, it is possible to overcome this issue: we have developed a second interface to the Webservice:

`tariqa3.0_service`; this interface is hosted on-, and only accessible from the main server.

Assessing that the server is properly maintained and updated; this mechanism ensures an equivalent level of security on the Web Service side than for remote authentication check; indeed only applications running from the server (which have been approved by the server manager) itself may access the service.

Please note that, meanwhile, full remote access to the Webservice is still available and protected by the Webservice's internal authentication mechanism usual authentication mechanism).

I.6.d, Caching

In addition **Tarîqa** use's a caching mechanisms in order to cut out the loading time of frequently/recently used contents.

To some extent (limited only to what is contractually allowed by each “Engine”), the caching mechanism is also applied to the loading of “Articles” and “Queries (i.e. The LexisNexis' Engine does not allow to store Queries' resultsets for more than 48h. Therefore, the articles' content are fetched and cached in a way that follow these rules.)

I.7. Authorization & Access levels

As many Engines in **Tarîqa** are commercial products, that are bound to a cost; and also as each

Engine has its own contractual policy with the institution (one may allow all employees to use it, another may only allow the permanent officials, a third one may only accept a limited group of users to access it); it was necessary, from the very beginning, to ensure that in no occasion an end-user would have the possibility to access an Article, Source or Engine she/he is not legally allowed to.

In addition, through **Tarîqa**, end-users have the opportunity to share information they have gathered or built; it is therefore very important as well to authenticate each individual using the system in order to avoid impersonation attempts.

I.7.a, Four Access levels

After studying the various contracts with the Engine, we have identified four distinct groups (levels) of users (the list is ordered so that the first one has the highest possible access level; the last one has the most limited access level):

1. Our division (close colleagues, working on similar fields)
2. Our institution (all the officials of all the divisions)
3. Our counterparts in the EU Member States (selected officials of Member States' ministries)
4. Other interlocutors in International NGO (selected individuals in NGOs)

When setting up your instance of **Tarîqa**, you'll discover that those levels have been associated to specific "aliases"; they are as follows (alias names have been chosen for historical reasons):

1. RELEX_TARIQARX (Access to all Engines)
2. RELEX_TARIQAEC (Access to most Engines)
3. RELEX_TARIQAEM (Access to several Engines)
4. RELEX_TARIQANG (Access to free Engines only)

Even-though, only these four groups are set in the current implementation of **Tarîqa** (Adding or modifying them would require programming expertise); you are totally free to decide which of your engine Engine would be made accessible by which group.

In a small structure, you could by example assign all your users to the RELEX_TARIQARX group (simply ignoring the three other groups), and therefore giving them all a similar level of access.

Or you could decide to dispatch your various Engines into the four groups, and assign the end-users

to each group according to your own internal rules.

I.7.b, And a fifth one

In addition to the above mentioned four groups, exists a fifth group, which is not at all related to the access of the Engines:

- The administrative group (alias: “TARIQA_ADMIN”)

This group contains only the users who have managing rights on the **Tarîqa** application itself.

Members of this groups have, by example access to the “weboffice⁵” of **Tarîqa**.

5 Management application, available at the following URL:

http://YOURTARIQASERVER/special_page.php?module=weboffice&

II. Prerequisites

Before installing **Tarîqa 3**, the following software's should be installed and properly configured:

1. Windows operating system⁶ (XP, Server, Vista, 2000)
2. Apache HTTP Server⁷ (Version 2.2 or higher)
3. PHP Hypertext Processing Language⁸ (Version 5.3 or higher)
 - The following extensions should be enabled:
 1. php_curl
 2. php_gd2
 3. php_ldap (*if using LDAP for authentication*)
 4. php_mbstring
 5. php_mysqli
 6. php_openssl
 7. php_soap
 8. php_xsl
 9. php_tidy
 - PEAR⁹ should be installed together with the following modules:
 1. Config
 2. XML_Parser
 3. XML_NITF
4. MySQL Database Server¹⁰ (Community Server version 5.1 or higher)
5. Microsoft .NET Framework¹¹ (Version 1,1 or higher)
6. Apache™ FOP¹² (Version 1,0 or higher)

If you don't need the PDF export function, you can disable it through the main configuration file (public/www/config/main_config.xml) by deleting the tag:
 config/briefcase/export/pdf

6 <http://www.microsoft.com/en-us/server-cloud/windows-server/default.aspx>

7 <http://httpd.apache.org/>

8 <http://www.php.net/>

9 <http://pear.php.net/>

10 <http://www.mysql.com/products/community/>

11 <http://www.microsoft.com/net>

12 <http://xmlgraphics.apache.org/fop/>

III. Installation

III.1. Steps

1. Create two new folders inside the DocumentRoot folder:
 - tariqa3.0
 - tariqa3.0_service
2. Create a third folder, in a location of your choice, outside the DocumentRoot folder; by example in your ServerRoot:

- ServerRoot/tariqapro_soap

3. Copy the content of

InstallationRoot/public/www/

to

DocumentRoot/tariqa3.0/

4. Copy the content of

InstallationRoot/private/www/

to

DocumentRoot/tariqa3.0_service/

5. Copy the content of

InstallationRoot/webservice/www/

to

ServerRoot/tariqapro_soap/

6. Install the following two Windows Services:

- mDownloader
- mAggregator

By running the installers located in:

InstallationRoot/dataservices/win32-binaries

7. Copy

InstallationRoot/dataservices/win32-binaries/udf_processes.dll

to

your MySQL Plugins Directory¹³.

¹³ <http://dev.mysql.com/doc/refman/5.1/en/install-plugin.html>

8. Create the database schemas by running the following SQL scripts¹⁴ on your Database Server (you must connect using an account with `root` privileges on the databases)

- `InstallationRoot/public/database-scripts/tariqa3.sql`
- `InstallationRoot/webservice/database-scripts/meta_common.sql`
- `InstallationRoot/webservice/database-scripts/rss_aggregator.sql`
- `InstallationRoot/webservice/database-scripts/soap.sql`
- `InstallationRoot/webservice/database-scripts/tariqapro.sql`

Note: the `meta_common` and `tariqapro` schemas must reside on the same server instance.

9. **Tarîqa** delegates tasks related to the news sources to its own SOAP `WebService`. In order to enable communication between the two, you have to register a set of `logins` and `passwords` in the `WebService's Database` and to register them in the configuration file of **Tarîqa**. Using your favorite SQL Editor¹⁵, insert at least four records in the `soap.users` table. Setting for each of them a different `accesslevel` (i.e.: 10, 20, 30, 40)

III.2. Application's configuration

1. Allow `mDownloader` to bind to the `tariqa3` database by setting:

`dbConnectionString`

in:

`mDownloaderRoot/mDownloaderService.exe.config`

2. Allow `mAggregator` to connect to the `rss_aggregator` database by setting:

`dbConnectionString`

in:

`mAggregatorRoot/mAggregatorService.exe.config`

3. Configure Apache HTTP Server to serve **Tarîqa** by adding to its configuration file:

- a `VirtualHost` directive for the **Tarîqa** Web Pages

Located in `DocumentRoot/tariqa3.0/`

- a `VirtualHost` directive for the **Tarîqa** Internal Pages

Located in `DocumentRoot/tariqa3.0_service/`

¹⁴ <http://dev.mysql.com/doc/>

¹⁵ <http://dev.mysql.com/doc/index-gui.html>

- a `VirtualHost` or `Alias` directive for the **Tariqa** SOAP Webservice

Located in `ServerRoot/tariqa3.0_service/`

A template configuration file can be found in:

`InstallationRoot/public/apache-config/vhost-tariqa.conf`

4. Configure the **Tariqa 3** application by setting all required parameters in:

- `DocumentRoot/tariqa3.0/config/main_config.xml`
- `DocumentRoot/tariqa3.0/config/news_config.xml`
- `DocumentRoot/tariqa3.0/config/soap_config.xml`
- `DocumentRoot/tariqa3.0_service/config/config.xml`
- `ServerRoot/tariqapro_soap/Tariqa/config/config.xml`
- `ServerRoot/tariqapro_soap/TariqaAuthentication/config/config.xml`

5. Enable SMTP e-mailing by setting:

- `EC_SMTP_SERVER`
- `EC_SMTP_SERVER_PORT`

in

`ServerRoot/tariqapro_soap//Tariqa/lib/ecsmtp.inc.php` (lines 35 & 41)

6. Configure the **Tariqa 3** Webservice by setting all required parameters

in:

- `tariqapro_soap/wshelper/lib/data/data_objects/fdk/config/factivaclient.xml`
- `tariqapro_soap/wshelper/lib/data/data_objects/wks/config/config.xml`
- `tariqapro_soap/Tariqa/fdk/config/factivaclient.xml`
- `tariqapro_soap/Tariqa/wks/config/config.xml`

7. Finally, restart the Apache HTTP Server's Windows Service

III.3. Webservice's configuration

III.3.a, Tariqa Webservice

1. Using your favorite text editor, open the file:

`webservices/www/Tariqa/Tariqa.wsdl`

2. Locate the text "`your-soap-address/`" and replace it with the URL used by your server to serve the Webservice (i.e.: <http://my-site.tariqa.com/soap/>)

Attention: Don't forget the final slash ("/") character!

3. Save the file and copy it inside the following two folders:
 - tariqa3.0/tariqa3/SOAP/wsdl
 - tariqapro_soap/tools/wshelper/wsdl

III.3.b, TariqaAuthentication Webservice

1. Using your favorite text editor, open the file:
webservices/www/Tariqa/TariqaAuthentication.wsdl
2. Locate the text "your-soap-address/" and replace it with the URL used by your server to serve the Webservice (i.e.: <http://my-site.tariqa.com/soap/>)

Attention: Don't forget the final slash ("/") character!

3. Save the file and copy it inside the following two folders:
 - tariqa3.0/tariqa3/SOAP/wsdl
 - tariqapro_soap/tools/wshelper/wsdl

IV. Customization

IV.1. Custom authentication mechanism

IV.1.a, Default mechanisms

Tarîqa has two default Authentication mechanisms that can be chosen from simply by selecting your favorite one in the Tarîqa configuration file:

ECAS/LDAP

Tarîqa supports an Authentication and Authorization mechanism based on ECAS¹⁶ and the institution's LDAP database, the first one provides the Authentication part, the second provides the Authorization part plus some additional data about the end-users.

This system is entirely managed at the institution level, and independently from Tarîqa.

Internal Database

By default, a very simple authentication & authorization mechanism is also provided by the means of a database table allowing storage of user information and access levels. This mechanism is implemented in the `tariqa3\tariqaAuthentication` (which is stored in the file:

`tariqa3/auth/tariqaAuthentication.class.php`)

To create a new user with this sytem, insert a new line in the `tariqa3.known_users` table.

By example:

```
insert into tariqa3.known_users values(
    'myid', null, 'n', 'n', 'My First Name', 'My Last Name',
    'mymail@mycompany.com', md5('mypassword'), 0, 'MY_GROUPS'
);
```

Where 'myid' is your desired user_id; mymail@mycompany.com is the desired e-mail address; 'mypassword' is the desired password; and 'MY_GROUPS' is the granted level of access.

MY_GROUPS can be any of the four groupsI.7.a ('RELEX_TARIQARX','RELEX_TARIQAEM',...)

or eventually one of the four groups plus the TARIQA_ADMIN group:

I.e.: 'RELEX_TARIQARX, TARIQA_ADMIN'

¹⁶ European Commission Authentication Service: <https://webgate.ec.europa.eu/cas/>

IV.1.b, Custom mechanism

Would you have your own Authentication mechanism already in place, or if the ECAS¹⁷ based mechanism was not capable of suiting your needs, please note that **Tariqa** also allow you to define your own authentication mechanism. For this purpose, two PHP classes must be developed:

iAuthentication/tariqaAuthentication

In order for **Tariqa** to recognize your mechanism, you have to write a new PHP class and name it:

```
tariqaAuthentication
```

that class must implement the following interface:

```
iAuthentication
```

the class should be saved in the following file:

```
ServerRoot/tariqa3.0/tariqa3/tariqaAuthentication.class.php
```

The code of the `tariqaAuthentication` class will depend on your own architecture. You'll find an example of a custom class in the following location:

```
InstallationRoot/tariqa3.0/tariqa3/tariqaAuthentication.class.php
```

This class implements custom authentication, based on an external application: ECAS (a closed system developed internally by the European Commission); but it can easily be adapted to work with any other mechanism.

iUser/tariqaUser

In order to present end-users to Tariqa, you will also have to write the following class:

```
tariqaUser
```

which must implement the following interface:

```
iUser
```

the class should be saved in the following file:

```
ServerRoot/tariqa3.0/tariqa3/tariqaUser.class.php
```

The code of the `tariqaUser` class will depend on your own architecture. You'll find an example of a custom class in the following location:

```
InstallationRoot/tariqa3.0/tariqa3/tariqaUser.class.php
```

IV.2. Content sharing & LDAP

The sharing function of **Tariqa** (which can be found on the Briefcase section), enables end-users to

¹⁷ European Commission Authentication Service: <https://webgate.ec.europa.eu/cas/>

share amongst each other Articles or Queries they have gathered through the system. They can also write their own texts and integrate them in the briefcase as a regular article.

In order to enable sharing, there must be a way to look for the system-users. In Tarîqa, that search is performed using an LDAP server, and users' tags. Each user may define a set of “Tags” (or keywords) and associate them to her/his own profile. This way it is also possible to look for users by their interest/expertise/.

Enable sharing by setting the value of:

```
/config/application/enable-share
```

To:

```
1
```

Configure the LDAP connection by updating accordingly:

```
tariqapro_soap/Tariqa/lib/ecldap.inc.php
```

In particular:

```
EC_LDAP_SERVER          (line 22)
```

```
EC_LDAP_SERVER_PORT     (line 62)
```

```
Base DN                 (line 30)
```

IV.3. Adding custom pages to Tarîqa: the “special pages”

As a simple way of extending **Tarîqa**, the system contains templates of so-called “Custom Pages” that can be fully rewritten by a web developer, and integrated into the application.

Pages written using that mechanism can benefit, with no efforts, from most functions of **Tarîqa** (Access to sources, information sharing mechanism, document repository, authentication/authorization, ...).

By example, Tarîqa contains a “Custom Page”, available at:

```
TARIQAURL/special_page.php?module=weboffice&
```

IV.3.a, Special page creation, step by step

For a fully working example, please see:

```
tariqa3.0/tariqa3/modules/special_pages/Ifs.class.php
```

1. Choose an identifier for the page.

i.e. "myNewPage".

2. Create a PHP class, and save it inside:

```
tariqa3.0/tariqa3/modules/special_pages
```

The class must:

Belong to the `tariqa3` namespace

Extend `SpecialPage` folder

3. Create a configuration file for the page inside:

```
tariqa3.0/config/special_pages
```

The configuration file must be named:

`PAGE_ID.xml` (where *PAGE_ID* should be replaced by the label chosen at step 1.)

An example configuration file can be found in:

```
tariqa3.0/config/special_pages/example.xml
```

4. Create a folder to store your page's content inside:

```
tariqa3.0/themes/default/modules/tariqa3/special_pages
```

The folder must be name:

`PAGE_ID` (where *PAGE_ID* should be replaced by the label chosen at step 1.)

5. Inside this folder, create:

1. a subfolder called:

```
styles
```

Inside which, create the CSS style file for your page. It must be named:

```
main.css
```

2. a template html file, that will contain the static part of your page;
name the file:

`PAGE_ID.html` (where *PAGE_ID* should be replaced by the label chosen at step 1.)

A full example can be found in:

```
tariqa3.0/themes/default/modules/tariqa3/special_pages/ifs/ifs.html
```

6. Register your page inside Tariqa, by adding the page's name to the list of pages:

1. "Include" your PHP class (at line 75) in :

```
tariqa3/special_page.php
```

2. Add the `class'` name to the list of modules (at line 92) in `tariqa3/special_page.php`
7. Register your page in the translation database:

1. Insert a “Section” in the ``languages_sections`` table:

```
INSERT INTO `tariqa3`.`languages_sections` (
  `section_id`, `description`
) VALUES (
  'module/PAGE_ID', NULL
);
```

(where PAGE_ID should be replaced by the label chosen at step 1.)

2. Insert the title of the section (it will be displayed in the Navigation Menu) in the database:

```
INSERT INTO `tariqa3`.`translation` (
  `section_id`, `keyword`, `description`
) VALUES (
  'module/PAGE_ID', 'tabTitle', NULL
);

INSERT INTO `tariqa3`.`translation_values` (
  `section_id`, `keyword`, `language_id`, `data`
) VALUES (
  'module/PAGE_ID', 'tabTitle', 'en', 'PAGE_TITLE'
);
```

(where PAGE_ID should be replaced by the label chosen at step 1. and PAGE_TITLE should be replaced by the label you want to display)

IV.3.b, Inherited Methods

getNews()

Inside your class, you may call the `getNews()` method in order to automatically insert the HTML code which is necessary to integrate a “News Section” (displays Articles provided by Query) on the page.

getDocumentRepository()

Inside your class, you may call the `getDocumentRepository()` method in order to automatically insert the HTML code which is necessary to integrate a “Document Repository” (allows upload/download of a folder's content) on the page.

IV.4. Adding an Engine to the system

1. Register the Engine in the Database:

```
INSERT INTO `tariqapro`.`engines` (
  `engine_id`, `name`, `data`, `access_level`
) VALUES (
  'ENGINE_ID', 'ENGINE_NAME', NULL, ENGINE_MIN_ACCESS_LEVEL
);
```

(Where `ENGINE_ID` should be replaced by the desired unique ID; `ENGINE_NAME` should be replaced by the desired display name of the Engine; `ENGINE_MIN_ACCESS_LEVEL` should be replaced by the ID number

2. Register the Sources in the Database (At least one Source must be registered)

```
INSERT INTO `tariqapro`.`sources` (
  `engine_id`, `title`, `source_data`
) VALUES (
  'ENGINE_ID', 'ENGINE_NAME', NULL
);
```

(Where `ENGINE_ID` should be replaced by the desired unique ID; `ENGINE_NAME` should be replaced by the desired display name of the Engine. The ``source_data`` column may be used to pass data to the Engine)

3. You may also already register some Queries for the new Source.

```
INSERT INTO `tariqapro`.`queries` (
  `source_id`, `title`, `query`, `time_range`
) VALUES (
  LAST_INSERT_ID(), 'QUERY_TITLE', '', 'previous(115)'
);
```

(Where `QUERY_TITLE` should be replaced by title of the Query. In the above example, please note that the syntax with `LAST_INSERT_ID()` does only work if the query is executed right after having inserte the Source in the database. In any other case, you will have to replace `LAST_INSERT_ID()` by the `source_id` of the Engine's Source you want to use.

4. Write the PHP class that will perform the actual Engine's tasks. That class must:

1. be named `ENGINE_IDEngine`
2. belong to the `tariqa3` namespace
3. extend `searchEngine`

4. be saved as :

```
tariqa3.0/tariqa3/search/ENGINE_IDEngine.class.php
```

(Where `ENGINE_ID` should be replaced by the Engine's ID, that was choosen on step 1)

5. define the following methods:

1. `listArticles()`

2. `multiAjaxSearch()`

3. `getSavePost()`

4. `getArticle()`

For examples, please see

```
tariqa3.0/tariqa3/search/MyNewSearchEngine.class.php
```

```
tariqa3.0/tariqa3/search/LexisNexisEngine.class.php
```

5. Register the Engine's internal Query Syntax with **Tarîqa**, so that it will be able to translate it from and to the **Tarîqa's Unique Query Syntax** (see I.5.d); to do so, update the file::

```
tariqa3.0/config/news_config.xml
```

For a detailed example, please see existing syntax translation inside:

```
tariqa3.0/config/news_config.xml.
```

IV.5. Adding an Export Format to the system

From the Briefcase section, Tarîqa permits exportation of Articles to various formats. Several formats are already available, such as PDF, MindManager/xMind, freemind mindmaps, CSV file, ...

The conversion system is based on XML/XSLT translations. Indeed, as the data processed inside **Tarîqa** are XML structured; it is extremely easy to transform them using XSL Stylesheets.

Here are the steps to register a new Export format:

1. Chose an (unique) Identifier for your format
2. Register it inside:

```
tariqa3.0/config/main_config.xml
```

By adding inside the tag:

```
config/briefcase/export
```

the following XML declaration:

```
<FORMAT_ID>
  <xsl>XSL_FILE_NAME</xsl>
  <content-type>OUTPUT_MIME_TYPE</content-type>
  <file-name>OUTPUT_FILE_NAME</file-name>
  <format>OUTPUT_HANDLER</format>
</FORMAT_ID>
```

Where:

- `FORMAT_ID` is the ID chosen at step 1,
- `XSL_FILE_NAME` is the name of your XSL stylesheet (without the .xsl extension)
- `OUTPUT_MIME_TYPE` is the Mime type that will be sent to the client's browser when she/he downloads the exported file
- `OUTPUT_FILE_NAME` is the default file name that will be sent to the client's browser when she/he downloads the exported file
- `OUTPUT_HANDLER` is the name of a PHP class that is capable of processing this specific type of output. The following handlers are already defined:
 - `txt` Handler for Textual formats
 - `pdf` Handler for PDF format
 - `mmap` Handler for MindManager(R) format

3. If the `OUTPUT_HANDLER` needed by your Export Format already exists, you may jump directly to the following point.

If it was not the case, you would first have to register a new `OUTPUT_HANDLER` in the system.

To do so, please follow these steps:

- i. Chose an (unique) Identifier for your handler (by example, the most common file extension for this type of output files.
- ii. The, inside:

```
tariqa3.0/tariqa3.0/tariqa3/briefcase_export
```

Create a new PHP class that:

- Is named `HANDLER_ID` (Where `HANDLER_ID` is the Identifier chosen in point i.)
- Is contained in a file called:

`HANDLER_ID.class.php`

(Where `HANDLER_ID` is the Identifier chosen in point i.)

- Has a default constructor with no arguments (or that provides default values for all arguments)
- Implements an “output” method, declared as follows:

```
/**
 * Perform the XSL transformation of some XML data and eventually
 * post process it
 * @param string $stylesheet Path to the XSL stylesheet
 * @param string $xmlstr The XML document to transform
 * @return string The transformed document
 */
public function output( $stylesheet, $xmlstr ) {}
```

For a detailed example, please see:

`tariqa3.0/tariqa3/briefcase_export/mmap.class.php`

4. Store your XSL Stylesheet as:

`tariqa3.0/themes/default/xsl/newsPrinter/exportFORMAT_ID.xml`

(Where `FORMAT_ID` is the Identifier chosen in point 1.)

Several examples can be found in the above mentioned location. The simplest one is:

`tariqa3.0/themes/default/xsl/newsPrinter/exportTXT.xml`

5. Finally register the format's name in **Tarîqa**, using the translation databases:

i. Register a key in the `translations` table:

```
INSERT INTO `tariqa3`.`translation` (
    `section_id`, `keyword`, `description`
) VALUES (
    'module/briefcase', 'export_briefcase_FORMAT_ID', NULL
);
```

(Where `FORMAT_ID` is the Identifier chosen in point 1.)

- ii. Register the text associated with this key:

```
INSERT INTO `tariqa3`.`translation_values` (  
    `section_id`, `keyword`, `language_id`, `data`  
) VALUES (  
    'module/briefcase', 'export_briefcase_FORMAT_ID',  
    'en', 'FORMAT_NAME'  
) ;
```

(Where *FORMAT_ID* is the Identifier chosen in point 1; and *FORMAT_NAME* is the name you would like to be displayed in the briefcase)