EXT: Single Sign-On

Extension Key: **naw_single_signon**

Extension version referenced in this document: 0.90

Table of Contents

Introduction

What does it do?

TYPO3 Single Sign-On (SSO) provides seamless integration of Third Party (i.e. non-TYPO3) Web Applications into the TYPO3 frontend. This includes

- access to “Third Party Applications” (TPAs) with no additional logon (for authenticated TYPO3-users),
- role-based integration of the TPAs into TYPO3 navigation or content,
- a sophisticated three-layer security architecture,
- no need for server-to-server communication, no need for central reverse proxies
- no need for a common/shared/synchronized password database or even user database.

This TYPO3 extension is the central part of the Signature-Based Single Sign-On Framework, but of course you need corresponding components on the remote (i.e. non-TYPO3) side.

Technically speaking: In order to integrate a Third Party Application (TPA), it needs to be adapted to the framework. This can be achived by

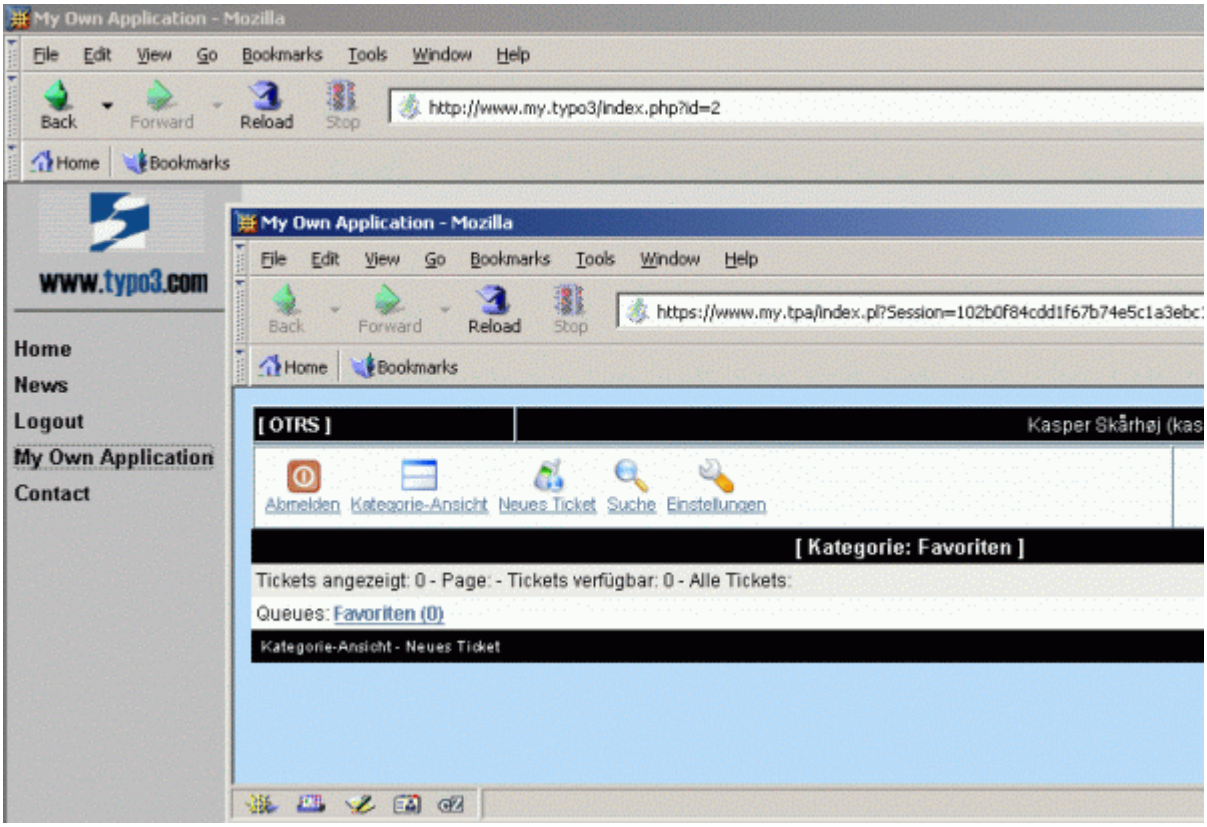
- using an existing “TPA Adapter” for your specific proprietary application - check the adapter repository: [1] TYPO3 Single Sign-On Homepage and Adapter Repository,
- using an existing TPA Adapter for an underlying generic trust-based authentication mechanism, e.g. for application servers like IBM Websphere or Bea Weblogic,
- creating your own adapter (this can be amazingly easy but depends completly on your TPA). Guidelines are given below.

In addition to the TPA Adapters, there may be more remote components, namely the so-called “SSO Agent”.

The remote components (SSO Agents, TPA Adapters) as well as the framework itself are not part of the TYPO3 extension and therefore not exactly in the scope of this document. Anyway, since they are mandatory for understanding and using the extension, some information is given in the appendix. You may even want to start reading there!

Screenshots

Picture 1 Frontend-User SSO view before login



Picture 2 Frontend-User SSO view after TYPO3 login and Third Party Application access (“open in new window” mode)

Administration

A Typical Scenario

Install and configure the TYPO3 Single Sign-On extension, if not yet done. See Installation and Configuration

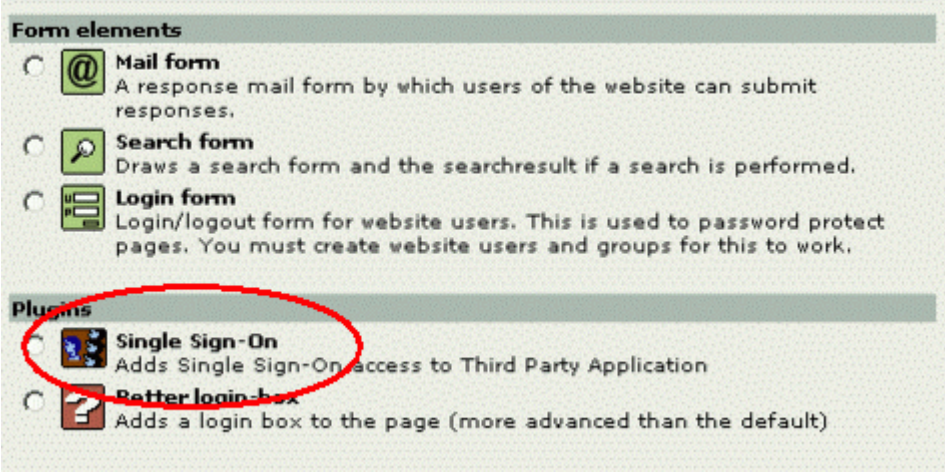
Install and configure the remote components (normally SSO Agent and TPA Adapter - this example's TPA_ID to configure in the SSO Agent config is "MyOwnApp"), if not yet done. See Installation and Configuration of SSO Agent and TPA Adapters

Now, there are several ways of integrating the SSO Link into your TYPO3 navigation or content. Here is a typical scenario:

- You want the TPA in the navigation menu, but only for authorized users.
- You want the TPA to open up automatically when selected from the navigation.
- At the same time, you want some explaination displayed in your content frame, plus a clickable link in case opening the TPA in a new window (through JavaScript) failed.

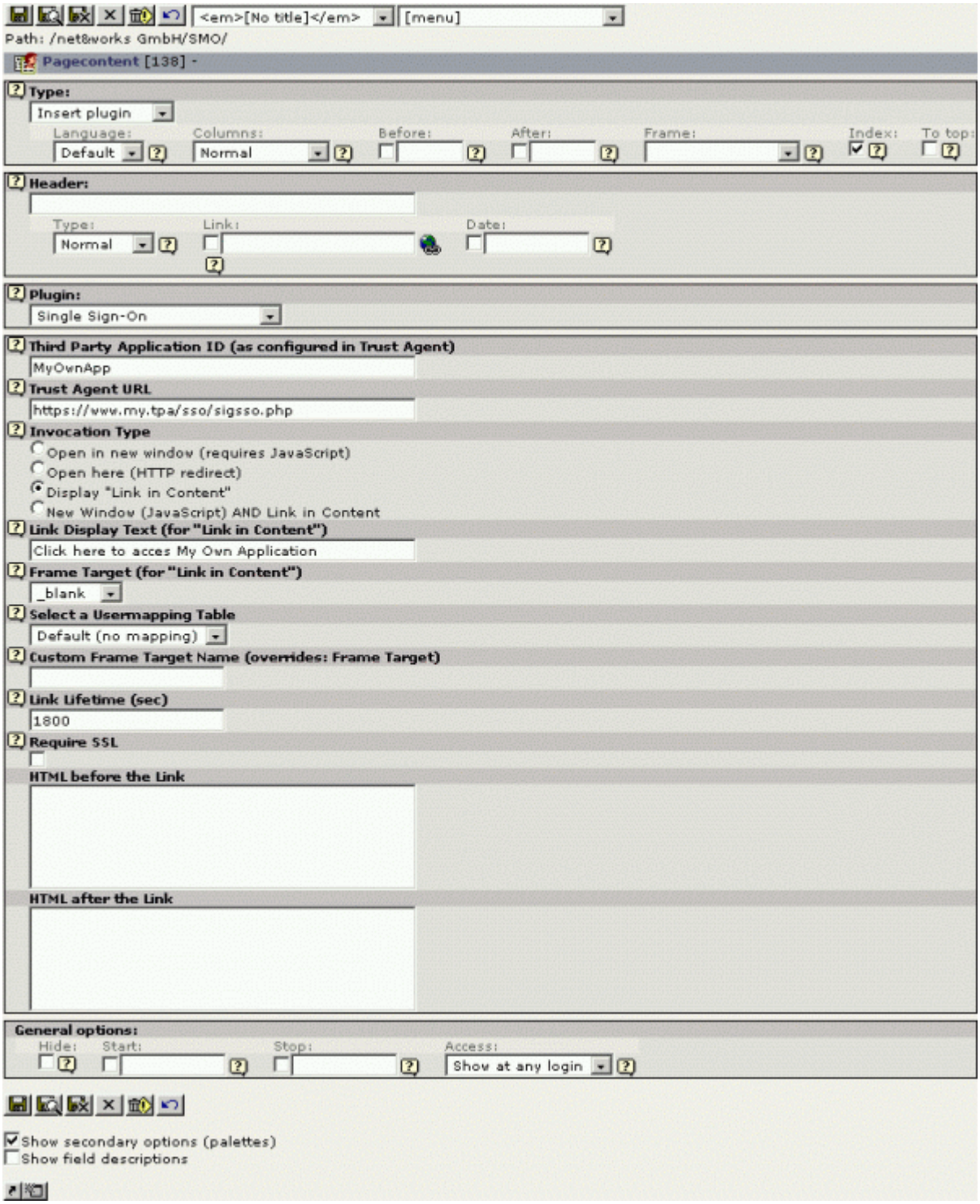
So what you might do is:

- Set up a new page in your tree with the desired display name for your TPA ("My Own Application").
- Limit the access to this page to the group of authorized users ("General Options" – "Access").
- Inside this page, create some regular content as usual. (Example: "My Own Application has been opened in a new browser window." And maybe a nice picture, too :-)



Picture 3 Insert the "Single Sign-On" plugin

- At a position of your choice, insert the "Single Sign-On" plugin
- and configure as follows:
    - Third Party Application ID: enter "MyOwnApp"
    - SSO Agent URL: enter "http://www.my.tpa/sso/sigsso.php"
    - Invocation Type: select "New Window (JavaScript) AND Link in Content"
    - Link Display Text: enter "Click here to access My Own Application"
    - Frame Target: select "_blank"
- Finally: Logon and try!



Picture 4 Single Sign-On in configuration options

Note: Before using this in a productive environment, make sure to read and understand at least the "Security Configuration Tasks" section !

SSO Content Element Parameter Reference

Field:	Explaination:
Third Party Application ID (as configured in SSO Agent)	A unique string that identifies the TPA. This TPA_ID is sent to the SSO Agent which calls the corresponding TPA Adapter based on its local configuration data.No blanks etc. allowed; case sensitive. Example: "MyOwnApp"
SSO Agent URL	The URL of the SSO Agent that handles this TPA, without parameters.Example: "http://www.my.tpa/sso/sigsso.php"

Field:	Explaination:
Invocation Type 1. Open in new window (requires JavaScript) 2. Open here (HTTP redirect) 3. Display "Link in Content" 4. New Window (JavaScript) AND Link in Content	What you want to happen when the users navigates to this page. 1. Immediatly open the TPA in a new window - requires Javascript (!) 2. Immediatly open the TPA in the present window (Frameset: in the content frame). This is done by HTTP redirect. 3. Just display a link (SSO Link to the TPA) in the page content. Configurable by additional options below. 4. Combination of 1. and 3.
Link Display Text (for "Link in Content")	Link Text to be shown in the content (if empty, the TPA_ID is displayed.) Example: "Click here to access My Own Application".
Frame Target (for "Link in Content")	Choose from standard targetsExample: "_blank"
Custom Frame Target Name (overrides: Frame Target)	Enter custom frame target (optional)
Select a Usermapping Table	If you configured User Mapping Rules (see below), here you can apply one to this SSO content element. Default is "Default (no mapping)"
Link Lifetime (sec)	For security reasons, the SSO Link has a limited lifetime. In case a user navigates around in TYPO3 up to a point where the SSO Link is generated, and then waits for a long time, the TPA access will fail with some "expired" message. Since for invocation types 1. and 2. the SSO Link is generated "on-the-fly", a rather short lifetime (like 10 secs) is sufficient in that case. For "Link in Content" invocation, you may want to allow some thinktime to the user. Default value (in seconds): 1800
Require SSL	If activated, the SSO Link will only be generated if the request method is HTTPS. Otherwise the content elements returns zero content. See Security Configuration Tasks on the importance of using SSL.
HTML before	Enter here HTML Code here before the SSO Link
HTML after	Enter here HTML Code here after the SSO Link

User Mapping Rules

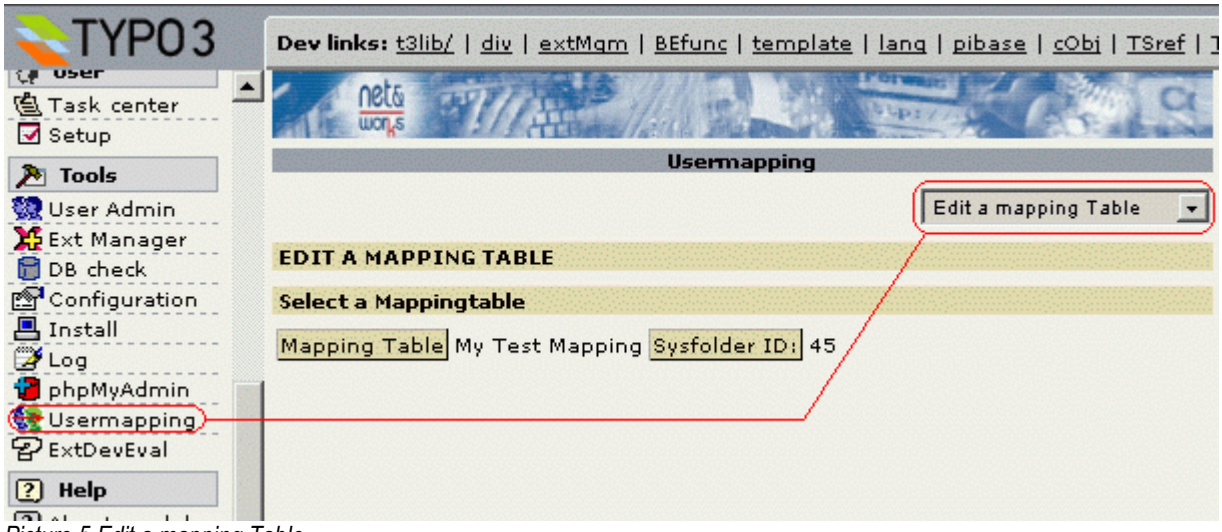
Eventually, you may find that the usernames in your TPA differ from the TYPO3 login names. In these cases you need to define User Mapping Rules that translate the TYPO3 to the name used in the desired TPA.

User Mapping Rules may be

- mapping tables that statically define "secondary" username strings for TYPO3 login names.
- pointers to external mapping data, e.g. in LDAP [not yet implemented]
- rule-based transformations ("change to uppercase", "add family name", ...) [not yet implemented]

User Mapping Rules are defined independently from the SSO content elements but can be assigned to them. Therefore one User Mapping Rule can be used for multiple SSO content elements.

User Mapping Tables



Picture 5 Edit a mapping Table

To work with User Mapping Tables, click "User Mapping" in the "Tools" section of the TYPO3 backend (Note: After freshly installing the extension, you will of course have to refresh the page in order to see the new option).

Here you can select from the following actions (using the drop-down menu):

- Info
- Create a new User Mapping Table
- Edit a User Mapping Table
- Delete a User Mapping Table
- Duplicate (copy) a User Mapping Table

Since a User Mapping Table is related to a TYPO3 user storage, you first have to select the proper sysfolder when creating a new table. For this purpose, a list of the existing sysfolders is presented to you. Choose one by clicking on it.

Inside the User Mapping Table, you can enter

- the table's name (this is the label displayed in the corresponding drop-down menu inside the content element configuration)
- and a "Mapped Name" per TYPO3 user.

Also you define the behaviour for users with an empty "Mapped Name": If the checkbox [currently: "map the username as it is"] is checked, these user names will be sent unchanged. If unchecked, for them access will be denied.

Configuration

Preparations

Prerequisites

- currently only tested on Unix Servers.
- currently only tested with Apache.
- currently only tested with TYPO3 3.5.0 and 3.6.0rc1.
- An RSA public/private key pair (see below)
- OpenSSL, preferrably compiled into PHP (otherwise the OpenSSL executable currently needs to be in the searchpath of the webserver's uid)

Note on OpenSSL: OpenSSL is used by PHP (not relatedto any usage by the web server), so for best results it should be compiled into PHP. Since often this is not the case, an option has been added that allows you to invoke OpenSSL externally. This option should be considered a way for quick testing but is depreciated in production environments. Also, the use of a passphrase is not supported in this case.

Creating RSA public/private key pair with OpenSSL

The entire TYPO3 Single Sign-On mechanism is based on the authenticity of the SSO Link. This is ensured by the TYPO3 server's OpenSSL signature, created and verified by OpenSSL. Therefore you need to create and deploy a public/private key pair.

Note: This is NOT related to the SSL keys you may be using for https in your web server !

In the following example OpenSSL will ask you to enter a passphrase. This is the OpenSSL private key Passphrase that you must enter for configuration when you install the Single Sign-On with the Extension Manager.

Example for usage with compiled-in OpenSSL:

Create a keypair:

- openssl genrsa -des3 -out sigsso_private.key 2048 [this will ask you to define a passphrase for your key ("Enter PEM pass phrase")]
- openssl rsa -pubout -in sigsso_private.key -out public.key

Optional you can remove the passphrase from your key with:

- openssl rsa -in sigsso_private.key -out sigsso_private_no_passphrase.key

Or here is an example to create a key pair without passphrase (i.e. for external openssl invocation):

- openssl genrsa -out sigsso_private.key 2048
- openssl rsa -pubout -in sigsso_private.key -out sigsso_public.key

Make sure to move the private key file ("sigsso_private.key") to a safe directory on your TYPO3 server, and minimize the access rights. Only the public key file ("sigsso_public.key") must be deployed to the SSO Agent machines. SeeSecurity Configuration Tasks for details.

Example:

- mv sigsso_private.key /usr/local/sigsso/etc/sigsso_private.key
- chown wwwrun /usr/local/sigsso/etc/sigsso_private.key
- chmod 400 /usr/local/sigsso/etc/sigsso_private.key
- scp public.key john@www.my.tpa:/usr/local/sigsso/etc/sigsso_public.key

(plus maybe a chown/chmod on www.my.tpa)

For more information on OpenSSL keys, see [4] Making OpenSSL public/private keys and Creating RSA public/private key pair with OpenSSL

Installation and Configuration

After installing the plugin through the EM, the first time you access the plugin module it will take you to the Configure Module option.

Here you will have to enter:

- Maximum displayed Users in the Backend while editing and creating Usermappings
- externalOpenssl (true or false) for use command line OpenSSL if you have PHP compiled without OpenSSL.
- Path to your private OpenSSL File
- Passphrase (if used internalOpenSSL function in PHP)

Note: Do not forget to submit this extension configuration form (click "update").

Extension Installation Screenshot

Picture 6 Installing the extension

Installation and Configuration of SSO Agent and TPA Adapters

SSO Agent

For installation and configuration, see the documentation that comes with your SSO Agent. A sample is given in this document's appendix.

TPA Adapter

For installation and configuration, see the documentation that comes with your TPA Adapter. A sample is given in this document's .

Security Considerations

Be warned: Implementing a Single Sign-On framework requires very careful planning and implementation, since it may potentially affect the security and safety of all your TPAs!

Design

Security has been the major objective throughout the development of the TYPO3 Single Sign-On extension. As of today, a carefully configured system is considered safe due to the following measures:

- No TPA passwords are stored in or even known to TYPO3 and the SSO Agents.
- The SSO Link (providing TPA access without password) has a limited lifetime (configurable).
- The SSO Link cannot be faked (it is signed by the TYPO3 server using an OpenSSL signature).
- Each SSO Link can only be used once (replay protection).
- Since the transmission of the SSO Link should be protected (otherwise it may potentially be wiretapped and used by an offender), the usage of SSL can be enforced by the TYPO3 extension.

A remaining objection may be that an administrator (a TYPO3 admin as well as the system administrator of the TPA machine) can find ways to logon to the TPA as another user, without knowing his password. This is true by design – but it should be added that this is true for many other applications even without TYPO3 Single Sign-On.

If you are going to create a very high security environment, you will find that any Single Sign-On solution is a tradeoff between security gains and losses, and you may even find that you are better off without single sign?on (see[3] Keys Botzum: "Single Sign On: A Contrarian View" for additional thoughts on this matter.)

Afterall, TYPO3 Single Sign-On provides a remarkably secure solution. And since both architecture and implementation are truly open, they are being examined (and, if necessary, improved) by a large community to make TYPO3 Single Sign-On as trustworthy as you need it.

Limitations

Remember that TYPO3 Single Sign-On provides no control about what a user is able to do inside a TPA or not. All you get is the user's identity, so you still need to

- set up each user inside each TPA,
- define each user's roles or rights inside each TPA,
- make sure that each TPA is free of exploits,
- think about implementing a firewall system on port filter level,
- think about implementing a firewall system on HTTP level.

Please be aware that logoff from TYPO3 does not include logoff from any TPAs.

For high security environments you may consider a commercial product (like Tivoli Access Manager) with central access control provided by a reverse proxy, terminating all HTTPS.

Security Configuration Tasks

a) TYPO3: Use SSL!

   The usage of HTTPS for TYPO3 Logon and espacially for the logged-on user (i.e. the session where the SSO Link is transferred) is highly recommended.

   Accessing the SSO Agent (sigsso.php) via SSL is also recommended but not mandatory.

   Consider using SSL for your TPA access, too (depends on the nature of your TPA).

b) TYPO3: Encrypt frontend user passwords!

   Use MD5 hashing for frontend user passwords (as described on TYPO3.org). Storing user passwords in clear is not acceptable for any serious environment.

c) TYPO3: Protect the plugin and its components!

Make sure only the designed SSO admins are allowed to access the global plugin configuration, the Single Sign-On content elements and most of all the User Mappings. Whoever can edit a TPA User Mapping, can also gain access to the TPA as any user desired. Thus, it has to be carefully protected from non-admin access.

d) TYPO3: Use user restriction for non-mapped TPAs!

If you do not have full control of the user administration on your TYPO3, make sure to use the "user restriction for non-mapped TPAs" feature. By doing so, you prevent others from just creating not-yet-existing TYPO3 users that correspond to existing TPA users.

e) TYPO3 system: Protect the OpenSSL Private Key file!

Since the signatur of the SSO Link' is a central part of the security concept, it is crucial to keep the OpenSSL private key file in a safe place. Obviously, it must not be accessible from the web.

f) TYPO3: Protect the OpenSSL Private Key passphrase!

As well as the OpenSSL Private Key file, also it's passphrase should be protected. It is configured in the extension setup, thus the access to this should be restricted.

g) TYPO3: Keep the entire system safe!

Since the security of the Single Sign-On relies on the integrity of the underlying system, all measures should be taken to keep this safe as well. Please follow the appropriate guidelines for TYPO3, mysql, PHP, Apache, Linux or whatever. There will be a seperate TYPO3 document (or maybe a section in the “Inside TYPO3” document) an this matter in the near future.

h) TYPO3: Consider system separation!

A security component like a Single Sign-On framework should ideally be as lean as possible, in order to reduce complexity and therefore bug or error probability. This is slightly contrary to the idea of integrating it to a system as complex as TYPO3. As an improvement, you may consider splitting up your TYPO3 environment into different systems.

i) Target System: Protect the TPA Adapter!

Make sure nobody but the SSO Agent (sigsso.php) (and root - maybe :-) can call the TPA adapter from the command line,

j) Target System: Protect the server instance running sigsso.php!

Since the web server serving the SSO Agent is able to execute sigsso.php and therefore generate valid user sessions, make sure to keep the risks to it low. In doubt, run a seperate httpd instance (under a different uid).

k) Target System: Protect the UsedTokens file!

Since the UsedTokens file (as configured in the SSO Agent's config) provides the replay prevention, fiddling it would be useful for an offender (in conjunction with snooping legal access to the SSO Agent). This appears to be a rather theoretical scenario, though.

Terms and Acronyms

one-time-token see "SSO Link"
SSO Single Sign-On
SSO Link the URL of the target system's SSO Agent including parameters (user, TPA_ID and verification data). Also referred to as the "one-time-token".
SSO Agent located on each target system, validates the incoming browser request (coming from the SSO Link) and invokes the matching TPA adapter.
TPA Third Party Application
TPA_ID TPA unique identification string, used to specify the target system of an SSO access
TPA Adapter A system call that creates a valid user session inside the TPA and returns all necessary information to the SSO Agent.

To-Do / Ideas for future directions

- Verify further OS'es / Web Servers / ...
- Redesign SSO Link generation: SSO content element does not contain SSO Link but points to another piece of code that sends the SSO Link via HTTP redirect (minimizes overhead, avoids expiration problems)
- introduce additional SSO Link parameter "Action=" (for future purposes like "activate SSO Link" or "Logout user")
- introduce additional SSO Link parameter "Version=" (for version compatibility)
- doubleclick protection or specific error message
- provide more adapters
- provide additional SSO Agent implementations (Java, ...)
- support for non-ascii characters (for uids, errorcodes etc.)

- provide adapters to other non-password based SSO systems (IBM: provide sigsso Trust Association Interceptor for Websphere et al.)
- provide adapters to password-based SSO systems (IBM: LTPA); provide option to auto-fill-in of logon forms of non-SSO enabled TPAs [Both would be a major change of design, since up to now we don't know or cache the user's password, by concept.]
- remote SSO Agent (not on the TPA machine)
- external user mapping
- remove users from user mapping when deleted from fe_users
- some protection for TPA Adapter system call
- Logout propagation
- clock offset management (Extension periodically sends time to SSO Agents, which calculates offset and adds this to the expiration values)
- Enhanced Security architecture: Extension activates SSO Link at the SSO Agent (optionally: including browser IP) prior to handing it to the browser
- give OpenSSL binary path
- include Server+Webpath into signature

Ressources

Additional Readings

[1] TYPO3 Single Sign-On Homepage and Adapter Repository

http://www.single-signon.com

[2] The Open Group: "Introduction to Single Sign-On"

http://www.opengroup.org/security/sso_intro.htm

[3] Keys Botzum: "Single Sign On: A Contrarian View"

http://www.software.ibm.com/wsdd/library/techarticles/0108_botzum/botzum.html

[4] Making OpenSSL public/private keys

http://www.openssl.org/docs/apps/openssl.html

[5] Creating public/private keys and certificates with OpenSSL

http://www.xs4all.nl/~dorus/linux/openssl.html

Commercial SSO products

Tivoli Access Manager

Netegrity SiteMinder

Appendix: Signature-Based Single Sign-On Framework

This is a general desription of the architectural framework and general definitions for Signature-Based Single Sign-On. Please see [1] TYPO3 Single Sign-On Homepage and Adapter Repository for the most recent version.

Architectural Overview

Signature-Based Single Sign-On does NOT rely on a central reverse proxy, on server-to-server communication etc. but allows direct access to the TPA by securely passing a one-time-token to the browser (via URL). Thus, TPAs may be distributed across the net.

Basically, we find a 3-layer architecture:

- TYPO3 (the Single Sign-On extension) dynamically creates a link ("SSO Link") that includes the desired TPA, user name, and security information (lifetime and signature).
- The SSO Agent, located on each target system (the machine where the TPA lives), validates the incoming browser request (coming from the SSO Link), talks to the TPA Adapter, and gives back an HTTP redirect to the browser that points to the TPA itself.

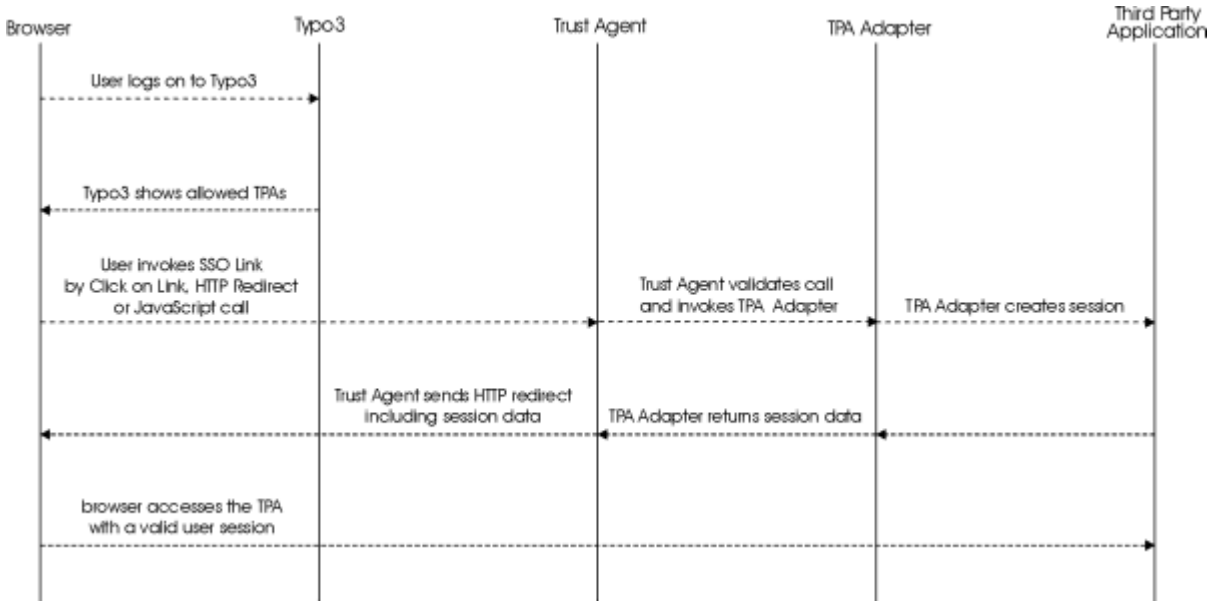
  Currently, the only SSO Agent implementation is in PHP script, but it can easily be ported to other languages.

  SSO Agents may be installed on multiple target system if you have TPAs on more then one server.

- The TPA Adapter is invoked by the SSO Agent (e.g. via PHP include or via system call). It creates a valid user session ("logs on the user") by application-specific means, and returns all information needed to the SSO Agent (in a defined format).

  This adapter is TPA-specific - this means that you need to find or develop an appropriate adapter for every TPA that you wish to integrate. It may be written in any language you favour. See the documentation that comes with your SSO Agent for details.

  See [1] TYPO3 Single Sign-On Homepage and Adapter Repository for a repository of existing TPA adapters.



Picture 7Signature-Based Single Sign-On: General Concept

One target server may optionally contain several TPAs, so each SSO Agent can be configured for more then one TPA adapter.

Options for Single Sign-On Integration

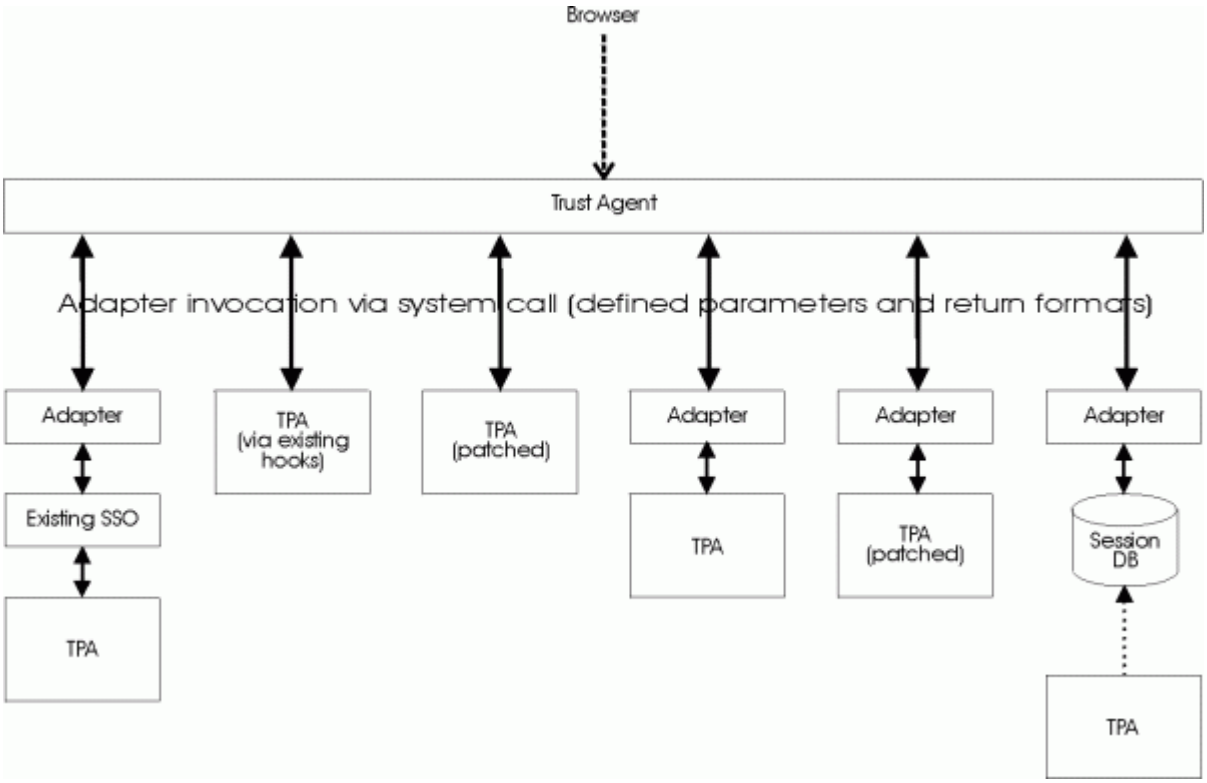
If you are going to integrate another TPA into your TYPO3 Single Sign-On environment, the first look will always be for existing TPA Adapters or generic ways like an adapter for an underlying Application Server (see What does it do?).

In case the is no such TPA Adapter available, you will need to create your own. The two major options you have are:

a) Use an existing single sign-on method supported by your TPA that is trust-based (i.e. does not require the user's password). Although this is rather uncommon, it is the recommended way if possible. What you effectively do is build an adapter to the single sign-on method supported by your TPA; the TPA itself is not affected.

b) Find a way to create a user session in your TPA. This may be done by hooks in your TPA, patching it, external code, by direct access to some session database etc. - it all depends on the nature of your TPA. Obviously, all this may become pretty tough if your TPA is not open source.

This leads to a considerable number of alternative variants; some of them are shown in the following picture.



As a convention, each of these variants is called "TPA Adapter", even there if is no dedicated (or no TPA-specific) adapter software.

In rare cases, implementations may make sense that do not seperate the SSO Agent functionality from the TPA adapter. These are called "Monolithic TPA Adapters".

Formats and Definitions

When developing own components, please make sure to follow these standards.

SSO Server --- SSO Agent

The SSO Agent is invoked via HTTP(S) access.

The SSO Server (TYPO3 extension) generates a link to the SSO Agent with the parameters

user	the user id that is to be logged on to the Third Party Application
tpa_id	the id configured in the SSO Agent that identifies the desired TPA
expires	the absolute time until when this link may be used (seconds from 1970)
signature	the RSA/3DES signature (using the private key configured) of the above keys and values as represented in the URL string. The order of the parameters must not be changed.

There may be more parameters may follow in upcoming versions.

Example:

```
https://my.tpa/sigsso.php?user=mytestuser&tpa_id=MyOwnApp&expires=1076925657&signature=9c79040a0a
cf28a3512a4bea6609dd2a31cbbf7421b77817f97162d051f03ee76729e3f98d26690a5bc7967f6e6c38f0454fe71a9603c6a126
d853f40b898721abf051a10b24100afd65458b95d2d5c37a4a5faa17c6be041caf028a9863049dcb15ae7c6cef47bfb171c47c50
b3424fa5d6660207b57d5e2737a76c6eb8837187b317e0171029a4705c30dc4c7dcc0716c797ae712df82be315814ef5da5b21d8
aeaf1537bdb16a6342bfbea64834853681d09461affa8ae09eb2388a104e1194141532cdc982ad9850ce6357065d8a1f5498f49a
e7ba30865ed5d1d2760332528fd1455c025f4bdf7702c83bd44dc839f5dc824d1582b024efb6a6a0aa3855
```

In PHP, the signature generation may read:

```
$this->data = 'user='.$user.'&amp;tpa_id='.$tpaId.'&amp;expires='.$expires;
[...]
openssl_sign($this->data, $this->signature, $this->pkeyid);
```

SSO Agent --- TPA Adapter (invocation: command line)

In this scenario, the TPA Adapter is invoked via system call.

Command line parameters are given for input values. Output values are to be returned on STDOUT.

Input values:

- all parameters must be preceded by a double dash (“--”)
- parameters that must be supported by an SSO Agent and may be used by a TPA Adapter are:
  remote_addr
  agent
  url
  user

Example:

/path/to/tpa/script --remote_addr=%remote% --agent=%agent% --url=https://redirect.url/index.php –user=%user% --moreparameters=anything_you _need

Output values:

The TPA Adapter must return a redirecturl.

The TPA Adapter can also return Cookie Parameters (see An example output of the TPA Adapter for two cookies may look like:. for the parametername s). Each parametername and value is separated by a whitespace (<tab> or <space>). Each pair in a new line.

Each cookie must start with 'CookieName'. If you need to set more than one Cookie you only need to send as many Cookie parameter sets as you need. The SSO Agent will notice that different Cookies are to be set.

An example output of the TPA Adapter for two cookies may look like:

redirecturl https://redirect.url/index.php
CookieName valuename1
CookieValue value1
CookieExpires expires1
CookiePath path1
CookieName valuename2
CookieValue value2
CookieExpires expires2
CookiePath path2
CookieDomain domain2
CookieSecure secure2

SSO Agent --- TPA Adapter (invocation: PHP method)

In this scenario, the TPA Adapter is invoked on the PHP level.

In your SSO Agent config, just give the PHP include file (php://path/to/tpa/script.php) and the redirect-URL.

Example:

/path/to/tpa/script.php –url=https://redirect.url/index.php

This will include the script and call the function sso($user,$remote_addess,$user_agent,$redirect_url) in the incuded script.

The included script needs to return an array with the following format :

```
Array(
         [redirecturl] => $redirecturl
         [0] => Array(
              "Cookiename" => $cookiename
              "CookieValue" => $cookievalue
              "CookieExpires" => $expires
              )
         [1] => Array(
              "Cookiename" => $cookiename
              ...
)
```

where “`redirecturl`” is mandatory (and may include session ID et al.), cookies are optional.

'[1..*] => Array' are arrays containing further cookies in case you need more than one cookie.

TPA Adapter Developement

To develop your own adapter, one way would to inspect the existing TPA code, find the point where user logon happens and patch this directly.

In the long run you may find it a better idea to duplicate that piece of code and patch this copy, since future upgrades of the TPA software won't affect the adapter (as long as the session handling is unchanged). You may also strip down the code to the necessary parts for the adapter.

Now, what exactly does "patching the code" mean here?

- The patched code create a new session for the given user without a password
- this session data will be stored in the TPA database
- The TPA Adapter must return a redirect URL where the browser should be redirected to.
- Optionally, the TPA Adapter can return cookie parameters (see above)

If your TPA is using the "Basic Authentication" mechanism, it may be a bit more tricky to adapt this - probably you will have to patch the actual application instead of just adding an external adapter.

Make sure that the TPA Adapter cannot be called from outside e.g. via web access. A way to do this is would be to place the file in a safe directory, or to check for the existence of environment parameters.

TPA Adapters - Enhanced Features

a) TPA "Change Password" Handling

   If you want to avoid some potential confusion, you should hide any existing "change password" functionality in your TPA from users that came in via SSO. Otherwise such a user may use this "change password" and expect the new password to work with TYPO3 (which it does not, since this "change password" only affects the TPA's own user registry).

   An even better way would be redirect the TPA's "change password" link to TYPO3's "change password", again: for SSO users only.

   Anyway: What you need to implement is

   · some flag in the user session that marks it as an "SSO-session" and

   · a switch that hides or changes the TPA's "change password" link for "SSO-sessions".

   [Of course... If you happen to have a nice environment with a common user/password repository (e.g. shared LDAP) in place, you will not need all this. Beyond the scope of this document, though...]

b) Policy Enforcement for Single Sign-On

   As another enhancement, you may want to check the user's role (i.e. group membership) during TPA Adapter sign-on. A possible policy would be to deny single sign-on access for administrators, another would be to restrict certain users or roles to special IP addresses or networks.