

Welcome to the Fast Seduction Developer Network (FSDN)!

*This file describes the **Community Connector** program of FSDN.*

About FSDN

FSDN was established to allow third party web developers opportunities to create resources and services geared for the community centered on the Fast Seduction 101 web site at FastSeduction.com.

Our goal is to put power in the hands of developers who have ideas they want to deploy very quickly to a large pre-qualified audience, with all registration and initial account management maintained by the FastSeduction.com web site.

This file will include any technical information you will need to make use of your approved developer account.

If you would like more information about FSDN policies or tiers (as well as commercial opportunities), please visit the following URL for the most up-to-date information:

<http://www.fastseduction.com/fsdn/>

If you have any questions about the information in this file, you can direct your queries to formhandle@fastseduction.com.



What Community Connector is

Community Connector is a means by which registered members of the community on FastSeduction.com (via their mASF account) can be securely bridged to services on third party web sites and securely returned to FastSeduction.com. No private data of members are shared with developers of such third party services, only the member name and/or unique identifier of that member such as member ID number. The user name and/or member number is the only element of the member's account which is passed to the third party. All other data passed is part of a secure exchange process to ensure trust between the services and that the username/ID passed has been validated by FastSeduction.com.

The system has been designed to be fully secure, establishing a method of bridging which involves encryption, backend authorization, and secure session tracking. We do all the hard work of making the bridge work and maintaining security of the process, leaving you to focus on designing services for an already existing large user base.

What Community Connector is not

Although some programming and design support may be provided to help developers implement Community Connector or make better use of it, FSDN is not a general development library, suite or API. It is primarily a secure bridging method to allow third party developers to offer services to members of the pickup and seduction community on FastSeduction.com.

Technical Requirements

- Have a web hosting account which supports CGI
- Your web host must support Perl Scripts (although you can use whatever programming language you want for your own services - our code is only there to bridge the users across sites)
- You must be hosted on your own domain
- You must be able to create a directory outside your public web paths and be able to read/write to it
- Your web-based programs which you want to use with Community Connector must be able to utilize cookies and maintain a secure session state for users

Basics

After being approved for a developer account with FSDN, you will be sent all the scripts and materials needed to deploy our Community Connector on your site/service. Your developer package for Community connector will contain a few files. You edit a couple settings in one of the files to match your environment and developer key you've been given. Then you upload the files to your /cgi-bin/. Then you just modify your login process to include a simple check against a couple of cookies which are generated into a user's browser when they've been bridged to your site. Those cookies, in combination with the bridging process, will allow you to be able to ensure the request is valid and complete the bridging process.

Installation & Deployment

In addition to this file, your developer package for Community connector will contain the following files:

```
fsdn.cgi
FSDN_Lib.pm
Blowfish_PP.pm
```

You will receive a developer name/key separately via email.

Open fsdn.cgi in a plain text editor. If you the line breaks (line feeds) seem messed up in the file then you need to find an editor which supports Unix line feed characters.

- Edit the **\$DEVNAME** variable to be the developer account name you've been provided. ***This MUST match what FastSeduction.com has on record for you.***
- Edit the **\$DEVKEY** variable to contain the developer key you were given. ***This MUST match what FastSeduction.com has on record for you.***
- Edit the **\$SESSIONPATH** variable to contain the exact/full path to a directory your CGI scripts can read/write to and also should be a directory safely outside your public web root.
- Edit the variable(s) in the **%LOGINBOUNCE** "hash" which will define the IDs & bridge URLs for any features/services you will be deploying with Community Connector - ***note that any IDs/features defined here need to be approved by FastSeduction.com, since we have to keep in sync with the IDs associated with your service offerings.***

The bridge URLs for features/services should contain a check which will validate the cookies passed by a bridged user. This is a very important check and SHOULD NOT be skipped. The check in your own code should run through the following logic:

1. Check for the existence of 2 cookies from your site visitors:

- **FSDNSESSION** and **FSDNUSER**
- FSDNSESSION is a string containing only alphanumeric characters (i.e. "2asksd8asms0asd")
- FSDNUSER is a string containing user name/ID in the form of "masf_#_USERNAME" where # is the user's mASF member ID and USERNAME is the user's mASF login name. You must run this cookie through an un-escape filter before using it.

2. If you see neither cookie then the request was not bridged. If you see one cookie but not the other then it's an invalid session. If you receive both cookies but one or both have empty values then it's an invalid session.
3. Ensure the FSDNSESSION cookie ONLY contains alphanumeric characters. If it contains any non-alphanumeric characters then the session is invalid.
4. Check for the existence of an associated session file (created by the Community Connector bridge) in your \$SESSIONPATH directory with the following syntax:
 - FSDNSESSION_fsdn.txt (where FSDNSESSION is the FSDNSESSION cookie). If the file does not exist then the session is invalid or expired.
 - Un-escape* any character-encoding of the FSDNUSER cookie, open the FSDNSESSION_fsdn.txt file and compare the contents to the content of the FSDNUSER cookie. If the match fails, then the session is invalid. If there is a match then the session is valid and you can trust that the # (Member ID) and USERNAME portions of the FSDNUSER cookie are valid and guaranteed to have been authenticated and come from FastSeduction.com.

* If you're unsure what character-encoding is, basically it's just reversing any URI-encoding which may have been applied to the cookie text, for example spaces might be escape to %20, brackets to %3C or %3E, etc and if that is the case will need to be de-encoded ("un-escaped"). mASF usernames don't contain the % character so it will always be safe to un-escape even if the FSDNUSER does not need to be, to simplify your coding.

After a bridge request has been validated, you can then bind the bridge to a login session on your end as if the person had logged in through your site. After that you can manage the session as you would any other login session with unique credentials on your site. To return the member/session to FastSeduction.com / mASF, you simply point them to the fsdn.cgi script on your site with the parameters "return=masf". If you want to provide visitors on your site a login for their existign mASF account, you can point them to:

[http://fastseduction.com/cgi-bin/fsdn/pass.cgi?action=login&p=**dev**&f=**feat**](http://fastseduction.com/cgi-bin/fsdn/pass.cgi?action=login&p=dev&f=feat)

Where "dev" is the developer account name you've been given and "feat" is the ID of the feature you which to have them bridged to.

It's that simple.

How to integrate bridged users to existing services:

Community Connector is viable primarily for services which would require a user to register an account, providing those users with features customized to their login.

NEW SERVICES: You can choose to design your service to not require any registration steps at all and simply depend on bridged connections/logins from FastSeduction.com or design them to allow for separate registrations in addition to supporting bridged connections/logins. The way you manage accounts registered directly through your site or ones bridged from FastSeduction.com and how to manage it all in a single user database is completely up to you. We recommend, however, that you read through the suggestions provided in this documentation to ease any issues you may run into related to account/username conflicts and avoid confusion for any users of your service.

EXISTING SERVICES: If you have an existing service, the best way we recommend bridging users is to determine how the service authenticates a login and have an intermediate script/program accept the bridge connection, generate any necessary cookies for the user session, then pass them to the service as if they'd logged in normally. For example:

Step 1: fsdn.cgi receives bridge and passes authenticated cookies to the user's browser, then redirects to your intermediate script bridgepass.php

Step 2: bridgepass.php does the last-step authentication step, and checks for the prior existence of the account wanting to be bridged. If no account exists, create one for the user dynamically and initialize a session for the user as if they'd logged in manually to the service, pass the session cookies of the service to the user's browser, then redirect them to the URL of the service. The process should be invisible to them.

User names:

We HIGHLY recommend NOT binding such sessions or user accounts to the "USERNAME" portion of the FSDNUSER cookie passed but rather the "masf_#" portion and add an extra tracking field (possibly "masfname" or whatever you want) to your user profile database specifically to track the username of the person who's been bridged into your service.

The first reason to do things that way is that members of mASF can change their username at any time but their member ID is the identifier which remains consistent. When they are bridged, you store their login as "masf_#" and use the "masfname" field to look up their current chosen name as the display name. If they change their username on mASF at a later time, bridges will still be successful, but you should compare the name they're coming in with and if it's different then update the "masfname" parameter in their profile with the new name passed before finalizing the session on your end.

The second reason to do this is to avoid conflicts with already-existing usernames of people who may have registered to the service using your own registration process. To avoid further conflict, and avoid problems with members being annoying to one another, retrofit your registration process to disallow anyone from manually registering usernames in the form `masf_#` (allowing only your own processes to handle such registrations at the time of initial bridging).

Password strategies:

No passwords of the member being bridged is ever sent to you. We consider the security of our members to be of greatest importance and therefore you cannot count on the member being able to log in with the same credentials as they would on mASF. In reality, given the way the bridging works, a password is not even necessary for the members and you can generate a dynamic password for them. We understand that some pre-existing services may offer the ability for members to email their password to themselves and in the case of bridged accounts, it is not necessary as their mASF login + bridge link will allow them secured access without needing to remember or even know the password assigned on your end (so make it random). It is up to you whether you want to provide them access to such a password or not but we recommend simply disabling access to it as it's not needed EXCEPT in the following case described in "Pre-existing accounts:".

Pre-existing accounts:

What to do if a pre-existing account name/login matches the login name of a bridged account (or visa-versa)?

To clarify, presume 2 different people wanted to access the services on your site.

In the first scenario, the first user registered an account called "MrFantastic". Later, a member from mASF who happens to have the username "MrFantastic" on mASF clicks on a bridge link and gets sent to your site. During the step after initial bridge validation, when you are checking account status, you can check to see if the name is already registered by someone else. In this case, the username "MrFantastic" is already registered to a pre-existing non-bridged user. The solution is to check for this possibility and present the user with a couple options. Tell them the name is already registered and if that is THEIR account (not their mASF account but an account they created separately on your site) then they can enter their password for that account, which allows you to "link" the bridge to that account OR if that username was NOT previously registered by them (indicating someone else already has that username) and offer them the opportunity to choose a different username for access. From that point on, any time they bridge into your service they will be identified by that name. Be careful to flag this somewhere in their profile or what will happen later is they will bridge back and your system will see their mASF username is different than what's on file and you don't want your system to automatically update their "masfname" field with the new name seen if there would continue to be a possible conflict with a username already in the database and you don't want them to be asked to choose a username every time they are bridged in.

In the second scenario, the user first accesses your service using a bridged connection. They are stored on your system using “masf_#” as their username and their mASF username stored in “masfname” in that profile (for your use only, not a field they can update themselves). Later, another user comes in and wants to register an account directly on your service with the same username as someone who previously bridged in. In addition to checking whether the username is already taken, also check the current “masfname” entries in your user database and consider any conflicts there as any other username conflict – tell them the username they want is already taken and have them choose a different name to register with.

Final complexity issue:

If a member who is initially bridged is forced to choose a different name in order to be bridged in (the case of a pre-existing account with that username, whether in the normal fields for username storage or in the “masfname” field described above), flag that account somehow as having been forced to choose an alternate name.

The reason for this is if you didn't do that, the next time the user bridges in from an mASF login, the system will find a match on a previously-bridged login but the name stored in “masfname” will not match. The system will then want to update “masfname” with the name being passed thinking the person recently changed their name on mASF. However, since that name previously had a conflict, they will be confronted AGAIN with the (forced) option to choose a different username since it's already taken. That would be quite annoying to that person since the point of Community Connector is to make the process as seamless and invisible to users as possible.

The way around this is to add a flag to the profile (like forcedname=1) to let you know the person had previously been forced to choose a different name and to not ask them again but rather just use the name automatically from the “masfname” field.

Avoiding any remaining potential username/login conflicts:

Beyond the above suggestions, we recommend that all new username registrations have the requested username wanted checked against your database in both the main username fields AND the “masfname” fields as described above, and don't forget to disallow registrations of usernames in the form of “masf_#”.

Troubleshooting:

Q: “I installed the script and libraries on my end but am getting a ‘500 Server Error’ whenever fsdn.cgi is accessed.”

A: This would usually mean that you’ve either entered the data for your variables incorrectly (there is a typo in the script), you’ve used a text editor which changed the Unix-style linefeed characters to Windows-style linefeeds, or you have not set the script to have execute permissions. Contact your web hosting tech support for help on making your script executable.

Q: “OK, I did all that but still get a ‘500 Server Error’.”

A: You might be missing some common Perl libraries. Ensure (by asking your web hosting tech support) that the following modules/libraries are available to Perl scripts running on your web site:

```
URI::Escape;  
LWP::UserAgent;
```

You should also check to make sure the first line of the script (referred to as the shebang line) points to the path where your perl interpreter is installed. For example:

```
#!/usr/bin/perl
```

Q: “Can I re-name the script from fsdn.cgi to something else?”

A: We don’t recommend it, but this is possible, you will just need to let us know what you’ve named the script so that the database on our end will know where to bridge people to.

Q: “I don’t know what a cgi-bin is and don’t know how to write web programs. This document is very confusing. Can I still use Community Connector somehow?”

A: Community Connector is designed for web developers who will be running web-based services. If you don’t know anything about web programming then you should not expect to be able to utilize Community Connector to offer services.

Questions & Answers about Community Connector:

Q: “Who is responsible for the services offered to users once they are bridged to me?”

A: You are. When they are bridged to you, you must treat them as people who have registered with your site because they were interested in your service. If they need support on your service, we will not provide that support since we don't know anything about your code, are not responsible for maintaining it, and don't have the time to handle support issues on services offered by other people. If the members we bridge to you encounter numerous ongoing issues and you are not properly handling their support questions then we will disable our bridge links to you and cancel your developer account.

Q: “I have a forum I want to use with Community Connector. Is that possible?”

A: Technically, it is possible, but we will not approve developer accounts for sites that host forums catering to pickup, seduction, dating, etc. The members we are bridging to you already have a forum available to them – mASF – and it makes no sense for us to bridge people to ... another form. The only considerations we will make is possibly for Lair sites with fewer than 250 members or sites with forums having little or no similar content to the type appearing on mASF.

Q: “What if I add a forum to my site after getting an approved FSDN account?”

A: We will review your site and if we feel it conflicts with our previously stated policies then we will contact you to remove the conflicting feature. If you don't comply, we will disable your developer account and remove all bridge links to your site. If you add something like that to your site without telling us, we will unilaterally cancel your developer account.

Q: “I have a question about what I am allowed or not allowed to do with Community Connector.”

A: Check the following URL for all our current policies:

<http://www.fastseduction.com/fsdn/>

Copyright Notice

Unless otherwise noted, all scripts and libraries and information provided to you is Copyright by Learn The Skills Corp, parent company of Fast Seduction 101 at FastSeduction.com. Redistribution of any materials, in whole or in part, without prior permission from Learn The Skills Corp is prohibited.

Trademark Notice

The following are registered or well-established trademarks of Learn The Skills Corp: "Fast Seduction 101", "Fast Seduction", "FastSeduction", "FastSeduction101", "FastSeduction 101". Use of these trademarks beyond what is already established within the scope of FSDN is prohibited without prior permission from Learn The Skills Corp.

Terms of Service Notice

Use of FSDN and related services must abide by our policies as described at:

<http://www.fastseduction.com/fsdn/>

Developer account access is made available to people solely at our discretion. Access to a developer account is not a guarantee and abuse of our policies will result in cancellation of the account and disabling of any supporting features.

FSDN is meant to be a cooperative venture for all parties involved and agreements made verbally or in writing are expected to be honored, else the relationship shall be deemed to be abusive.