# X7 Chat Simple Integration Guide

This document explains how X7 Chat’s authentication check can be integrated with an external software system’s user database (referred to as the “remote system” in this document).

## Basic Architecture

X7 Chat is embedded in remote web pages using an iframe. The iframe src attribute points at the X7 Chat index.php page.

Integrating X7 Chat with a remote system requires the implementation of two components:

1. An X7 Chat integration module
2. A module/plugin/extension/modification/etc. for the remote system (referred to as the “remote module” in this document)

These two components send encrypted messages between each other using HTTP. Three types of messages are sent:

1. An authentication check message: the X7 Chat integration module sends a request to the remote module containing a username and password, the remote module determines whether the authentication details are correct and returns an appropriate response.
2. A user details lookup message: the X7 Chat integration modules sends a request to the remote module containing a user ID, and the remote module response with account details for that user.
3. A session continuation message: when a user accesses X7 Chat from the remote system, the remote module may send X7 Chat a message containing the user’s user ID. X7 Chat will use this information to authenticate the user automatically, without the user needing to supply their username and password. This message is passed through the iframe src attribute; not via a direct API request between the two systems.

Additionally, the remote module is responsible for embedding the X7 Chat iframe in a remote page.

## The X7 Chat Integration Module

X7 Chat Integration Modules are placed in the includes/integration directory of the X7 Chat installation. The X7 Chat Integration Module consists of a directory named for the remote system. Inside that directory there must be an authenticator.php file and a config\_loader.php file. For example, the authenticator.php file for WordPress is located at includes/integration/wordpress/authenticator.php.

The authenticator.php file must declare a class named ‘authenticator’ that extends \x7\integration\remote\_authenticator. The ‘authenticator’ class must belong to the x7\integration\<remote system name> namespace (where <remote system name> is the name of the directory containing the authenticator.php file).

The authenticator class must declare a get\_url() method, which returns a string containing the URL of the remote module.

Note: If the ‘auth\_api\_endpoint’ configuration parameter is defined in X7 Chat’s config.php file, it will be used instead of get\_url(). For one-off implementations, the best option may be to return an empty string from get\_url() and simply rely on manually setting the API endpoint URL in the X7 Chat config.php file instead.

The config\_loader.php file must return an array containing configuration parameters for X7 Chat (parameters normally defined in config.php). If you do not wish to pull configuration details from the remote system automatically, then this file can return an empty array.

## The Remote Module

The remote modules included with X7 Chat are located in the integration directory; however, placing remote modules in this location is not a requirement.

The remote module requires 3 files and one third party library (phpseclib) included with PHP. These four resources must be present on the remote system, but the remainder of the X7 Chat code is not required.

When the remote module detects an incoming API request from X7 Chat, it must perform several actions:

1. Load the required resources:
   1. includes/model.php
   2. includes/model/api\_message.php
   3. includes/libraries/phpseclib/Crypt/Hash.php
   4. includes/libraries/phpseclib/Crypt/AES.php
   5. includes/api.php
2. Load or define the API key (in X7 Chat’s config.php file, this is defined as ‘api\_key’).
3. Construct an instance of the \x7\api class and pass the API key to its constructor:
   1. $api = new \x7\api($api\_key);
4. Decrypt the incoming message. The incoming message is POSTed as ‘message’:
   1. $api\_message = $api->get\_message($\_POST[‘message’]);
5. Handle the message appropriately.
6. Generate and output an encrypted response:
   1. echo $api->create\_message(new \x7\model\api\_message(array(‘key1’ => ‘value1’, ‘key2’ => ‘value2’)));

## Handling Authentication Check Messages

For an authentication check message, the $api\_message->method property will be set to ‘authenticate’. $api\_message will also have properties for ‘username’ and ‘password’.

If authentication fails, the API expects $api\_response->ok to be false.

If authentication succeeds, the API expects $api\_response ->ok to be true.

Regardless of whether authentication fails or not, if ‘username’ belongs to a valid user, then the API expects $api\_response ->user to be defined as an array with three keys: ‘id’, ‘username’, and ‘email’ (with appropriate values). If ‘username’ does not belong to a valid user, then $api\_response->user should not be set.

## Handling User Details Lookup Messages

For a user details lookup message, the $api\_message->method property will be set to ‘user\_details’. $api\_message will also a property for ‘id’.

If the user ID is invalid, the API expects $api\_response->ok to be false.

If the user ID is valid, the API expects $api\_response ->ok to be true, and expects $api\_response ->user to be defined as an array with three keys: ‘id’, ‘username’, and ‘email’ (with appropriate values).

## Session Continuation Messages

The session continuation message is an encrypted string sent in the URL to the X7 Chat index.php file. This is used to transfer the user’s active session from the remote system to X7 Chat. Often, X7 Chat is embedded in an iframe, so the session continuation message is part of the iframe’s src attribute.

To generate a session continuation message, construct an array containing the user’s user ID and pass it through the create\_message method of \x7\api:

$data = array(‘id’ => 5);  
$message = $api->create\_message(new \x7\model\api\_message($data));

The value in $message should now be passed in the query variable ‘session\_key’. For example:

http://x7chat.com/chat/index.php?session\_key=<value of $message>

Note that this message only remains valid for 14 seconds.