Using the Authentication Filter Tool in NetIQ® CloudAccess

Technical Reference

November 2015

This document provides information about using the Authentication Filter tool for NetIQ CloudAccess to make session-based changes to the identity information for an authenticated user.

The CloudAccess single sign-on login is designed to authenticate a user against an identity source and to share this authentication with other protected applications. The authentication process does not provide extended functions to add, remove, or manage a user's identity information for the session. To address this need, CloudAccess provides the Authentication Filter tool.

The Authentication Filter integrates with the CloudAccess single sign-on process. After the user logs in, the filter intercepts the authentication process and sends the user's identity information from the identity source to your custom authentication scripts. You can add, remove, or set values for supported identity attributes. You can also set a cookie. You can interact with the user to gather input for those changes. After all of the encoded rules and associations are complete, CloudAccess stores the modified identity information in the session cache for the web services and applications.

The Authentication Filter tool is compatible with the ExtAPI library and the ExtUI library. It works with multiple scripting languages including PHP, Java, and Perl.

Contents

Using an Authentication Filter

NetIQ CloudAccess uses information in identity sources to authenticate users. It retrieves identity attributes for an authenticated user, and shares that information with web services and applications during the user’s session. In some cases, this identity information might be insufficient to access all of your protected resources. Some applications might require confidential user information. Others might require information that only the user can provide. Still others might require special settings for identity attributes. You also might want to remove unneeded identity attributes for a user’s session to reduce the amount of information injected in query strings, headers, and assertions.

The Authentication Filter allows you to interrupt the login process to modify an authenticated user’s identity attributes for a session. You create external filter scripts that retrieve confidential identity information from a variety of back-end databases, to collect private information from the user, or to modify the attributes and their values based on authentication logic that you apply against the user’s identity attributes. You can gather information that is needed by one or any combination of the protected web services or applications that use session-based protocols. During the session, the appliance sends the user’s modified identity attributes to destination web services or applications instead of the attributes it retrieved from the user’s identity source.

To use the filter effectively, you must understand the type of identity information needed to access your protected web services and applications. For example, a medical service provider might need a user’s medicare insurance ID. A travel application might need a user’s passport number or VISA number. A social website might need a user’s loyalty account number.

You should determine whether the information is available in your identity source, if it is available from other enterprise databases, or if can be derived from information therein. For example, the following sensitive data is typically not found in an identity source, but might be stored in an enterprise database: Social Security Numbers; national ID numbers; passport numbers; VISA permit data; state driver's license number or non-driver identification number; credit card number, debit card number, or other financial account number; loyalty account numbers, and protected health information. If the information is not present, or you cannot access the databases for this purpose, you can create a user interaction script to gather the information directly from the user.

By using the Authentication Filter, you can customize a user’s identity attributes for a session without disturbing the attributes and values stored in your identity source. It gives you the flexibility to support a variety of web services and applications, including your in-house or custom protected resources.

How the Authentication Filter Works

The Authentication Filter integrates with the CloudAccess single sign-on process. It allows you to modify an authenticated user’s identity information as part of the login process, and then uses that modified information for the life of the session as the user accesses web services and applications.

Before you enable the filter, you must encode custom authentication logic that accepts an authenticated user’s identity information, and modifies it for use in the user’s current session. In the filter configuration, you specify the URL of your external program, as well as the Basic Authentication credentials to access the file, if required.

To begin a session, the user logs in as usual by providing credentials on the CloudAccess login page. Typically, the user provides a user name and password, or an email address and password, depending on the requirements of the user's identity source. If you configure additional CloudAccess security measures for authentication, the user must also successfully complete them. For example, you might configure CloudAccess to require two-factor authentication (such as time-based one-time password), or to issue a Google reCAPTCHA challenge.

After the user logs in successfully to CloudAccess, the Authentication Filter intercepts the session and sends the user's identity information from the identity source to the URL for the program that runs your custom authentication logic.

Your ExtAPI script uses the ExtAPI library commands to manipulate the user's identity information against the authentication logic that you encode, according to your business needs. You can add, remove, or set values for supported identity attributes. You can also set a cookie. The changes apply only for the user's current session. For example, you might set an attribute that allows the user to interact as a secure but anonymous user during the session.

If a change requires user input, you can redirect the session to another URL with an ExtUI script that uses the ExtUI library commands to interact with the user, and then return to CloudAccess. This allows you to collect identity information to use for the session. For example, if the user logs in with a LinkedIn identity, you can collect the user's loyalty award program account number for that identity. The Authentication Filter checks the user's input against your ExtAPI script to ensure that it completes all of its encoded rules and associations.

When the external authentication process is complete, the script returns information to the Authentication Filter about the modified identity attributes for the session, as well as the HTTP return status. The filter stores the user's session-based identity information in memory; it does not save the changes to the user's identity source. When the filter tasks are complete, the authentication process returns control to the user's browser.

When the user accesses SaaS web services and applications, CloudAccess uses the modified identity information to establish connections with them. CloudAccess deletes the user's modified identity information from memory when the user's session ends.

Creating Custom Scripts

To use the Authentication Filter tool, you must create custom scripts that execute extended authentication logic against the authenticated user's identity information. Use the information in this section as a guide for encoding the actions to meet your business needs.

- "ExtAPI Script" on page 3
- "Redirect and ExtUI Scripts" on page 4
- "Supported Scripting Languages" on page 4
- "Supported Identity Attributes" on page 4
- "JSON Requests and Responses" on page 5
- "Return Values" on page 5

ExtAPI Script

To use the Authentication Filter tool, you must create a script that uses the ExtAPI library commands to add, remove, or set values for supported identity attributes. You create an authentication script, such as `index.php`, and store it on the ExtAPI server. For example:

```
http://extapi_server_dns/extapi/index.php
```

When all encoded rules and associations are complete, the script returns the HTTP 200 OK response to CloudAccess.

For more information, see "Example: Setting Values and Removing Identity Attributes" on page 7 and "Example: ExtAPI Script" on page 11.

Redirect and ExtUI Scripts

If a change requires user input, you can redirect the session to another URL with a script that uses the ExtUI library commands to interact with the user, and then return to CloudAccess. The Authentication Filter automatically checks the input against your ExtAPI script to ensure that it completes all of its encoded rules and associations.

Create a redirect script, such as `redir.php`, and store it on the ExtAPI server. For example:

```
http://extapi_server_dns/extapi/redir.php
```

For more information, see "Example: Adding and Removing Identity Attributes with User Interaction" on page 8 and "Example: Redirect Script" on page 11.

Create a user interaction script, such as `user-input.php`, and store it on the ExtUI server. For example:

```
http://extui_server_dns/extui/user-input.php
```

Supported Scripting Languages

The Authentication Filter is compatible with many scripting languages, including PHP, Java, and Perl.

The Authentication Filter uses a JSON POST to send the authenticated user's identity information to your custom authentication script. The script uses JSON to send the HTTP return status and the user's modified identity attributes to CloudAccess for the session.

Supported Identity Attributes

The Authentication Filter supports the following identity attributes. Attribute names are case sensitive.

`ID` (read only; cannot be modified)

`UserName` (read only; cannot be modified)

`FirstName` (read only; cannot be modified)

`MiddleName`

`LastName` (read only; cannot be modified)

`FullName`

`PreferredName`

`GenerationalQualifier`

`Gender`

`Email`

`Phone`

`Photo`

`BirthDate`

`StreetAddress`

`City`

`State`

`ZipCode`

```
Country
Language
IdentityType
XCustom1
XCustom2
XCustom3
XCustom4
XCustom5
```

JSON Requests and Responses

The JSON request file contains the following configurable information:

- Identity attributes
- Actions to take (add, remove, set values)
- (Conditional) If the authentication action requires user interaction, the redirect URL to the ExtUI location
- Return to the appliance
- HTTP response code

For more information about the request format, see the “Example: JSON Request” on page 10.

The response file specifies the following:

- The values to set for specified variables. You can modify values only for the identity attributes sent in the request.

For more information about the response format, see the “Example: JSON Response” on page 11.

IMPORTANT: The ExtUI processes only information passed via the redirect location URI. The client can view and modify any information that is passed in the URI. Therefore, it is imperative for the security of the system that you encrypt the information so that only ExtAPI and ExtUI can decrypt it.

Return Values

The following HTTP response values are valid return codes. All other HTTP response codes result in a failed authentication.

HTTP 200 OK

The request succeeded. CloudAccess processes the specified changes and returns to the user.

HTTP 301 Moved Permanently

When the response returns a 301 value with a valid Location header, CloudAccess processes the changes and then directs the user’s browser to the new permanent URI. Any future references to this resource should use the URI as well.

HTTP 302 Redirect

When the response returns a 302 value with a valid Location header, CloudAccess processes the changes and then redirects the user’s browser to the specified location. The appliance does not consider authentication to be complete unless the user’s browser is subsequently redirected to the Location specified in the ReturnURL value from the original JSON request.

Configuring the Authentication Filter

After you create your custom scripts, you must enable and configure the Authentication Filter tool in CloudAccess. The enabled filter automatically runs on each node in a CloudAccess cluster.

Before you enable the Authentication Filter, ensure that your enterprise environment meets the following requirements:

- A CloudAccess appliance, installed and configured.
- The Authentication Filter supports only applications and devices that use session-based protocols. The filter stores the altered identity attributes and values in the session attribute cache.

  The Authentication Filter does not support applications and devices that use sessionless protocols, because there is no session attribute cache to store the altered identity attributes and values. For example:

  - The OAuth protocol is a sessionless protocol. Thus, the Authentication Filter does not support applications use the OAuth Service Provider connector.
  - Mobile devices use a token-based protocol, which reestablishes the session for each transmission. Thus, there is no session attribute cache for mobile sessions, whether the connector's protocol is session-based or sessionless.
- On the ExtAPI server, create a script that uses the ExtAPI library commands to apply session-based authentication rules to an authenticated user's identity information. The Authentication Filter points to the URL for this file. The ExtAPI server is a web server that supports the programming language for the script file you create.
- If the session-based identity changes require user interaction:
  - On the ExtUI server, create a script that uses the ExtUI library commands to collect the user's session-based identity information, and return control to CloudAccess. The ExtAPI script should redirect the authentication session to the URL for this file. The ExtUI server is a web server that supports the programming language for the script file you create.
  - On the ExtAPI server, create a redirect file configured with the ExtUI script's URL.

To enable the Authentication Filter:

1. Log in with an appliance administrator account to the CloudAccess administration console at `https://`*`appliance_dns`*`/appliance/index.html`.
2. Drag the **Authentication Filter** icon from the **Tools** palette and drop it in the **Tools** panel.
3. In the **Tools** panel, click the **Authentication Filter** icon, then click **Configure**.
4. In the Edit External Filter window, complete the following information:

   Display name: Specify a name for the filter. This name appears on the main Admin page.

   Connects to: Specify the URL to the script that you want to run during the user SSO login.

   For example:

   ```
   https://extapi_server_dns:port/path/extapi/index.php
   ```

   Use HTTPS for secure SSL transfer of information. If you use an HTTP URL, information is not secure.

   Basic Auth User: (Optional) If login is required to access the URL, specify the user name to use in the basic authentication header.

   Basic Auth Password: (Conditional) If you specify a user name, specify the password for it.
5. Click **OK** to save and enable the filter settings.
6. On the Admin page, click **Apply** to activate the filter configuration.

7 Wait while the service is activated across all nodes in the cluster. Do not attempt other configuration actions until the activation completes successfully.

In the **Appliances** panel, a green gear icon spins on top of each node until the activation is complete across all nodes in the cluster. In the **Tools** panel, a green status icon appears on the lower-left corner of the service icon. A yellow status icon appears if the URL uses HTTP instead of HTTPS because the traffic is not secure.

Examples

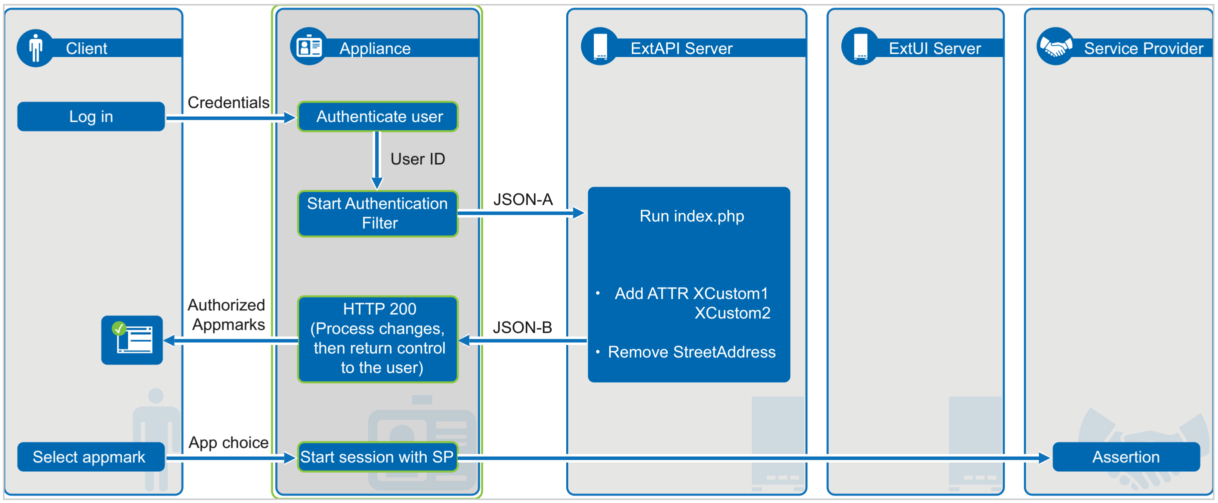
You can use the examples in this section to understand how the Authentication Filter interacts with your custom scripts to modify the authenticated user's identity information for a session.

- "Example: Setting Values and Removing Identity Attributes" on page 7
- "Example: Adding and Removing Identity Attributes with User Interaction" on page 8
- "Example: JSON Request" on page 10
- "Example: JSON Response" on page 11
- "Example: ExtAPI Script" on page 11
- "Example: Redirect Script" on page 11

Example: Setting Values and Removing Identity Attributes

Your custom authentication scripts can add, remove, or set values for the supported identity attributes. Because the changes do not require user interaction, the user is unaware of the external authentication tasks being performed. After the external authentication actions and associations complete successfully, the user can access the SaaS web service or application. CloudAccess establishes the session by sending the modified identity information. For an example of how to add identity attributes, to set values for new or existing identity attributes, and to remove an identity attribute, see the "Example: ExtAPI Script" on page 11.

Using the Authentication Filter to Add or Remove Identity Attributes



To prepare the Authentication Filter, do the following as an administrator user:

1. Create an authentication script, such as `index.php`, and store it on the ExtAPI server. For example:

`http://`*`extapi_server_dns`*`/extapi/index.php`

2. Enable the Authentication Filter tool in CloudAccess, and configure it to point to the target authentication script.

The extended authentication process is as follows:

1. The user logs in to the CloudAccess login page with the single sign-on login credentials.
2. CloudAccess authenticates the user credentials against the identity source.
3. The Authentication Filter intercepts the authentication process and sends a JSON POST that contains the user’s identity information to the URL of a custom authentication script on the ExtAPI server.
4. The script sets values for the attributes XCustom1 and XCustom2, and removes the attribute StreetAddress.
5. After all external authentication is complete, the authentication script returns a JSON response to CloudAccess that contains an HTTP return status and the user’s modified identity attributes for the session.
6. CloudAccess stores the modified identity information in memory; it does not store it in the user’s identity source.
7. CloudAccess completes the authentication process and returns control to the user’s web browser.
8. For the life of this session, the modified identity attributes are available to service providers when the user accesses a SaaS web service or application.

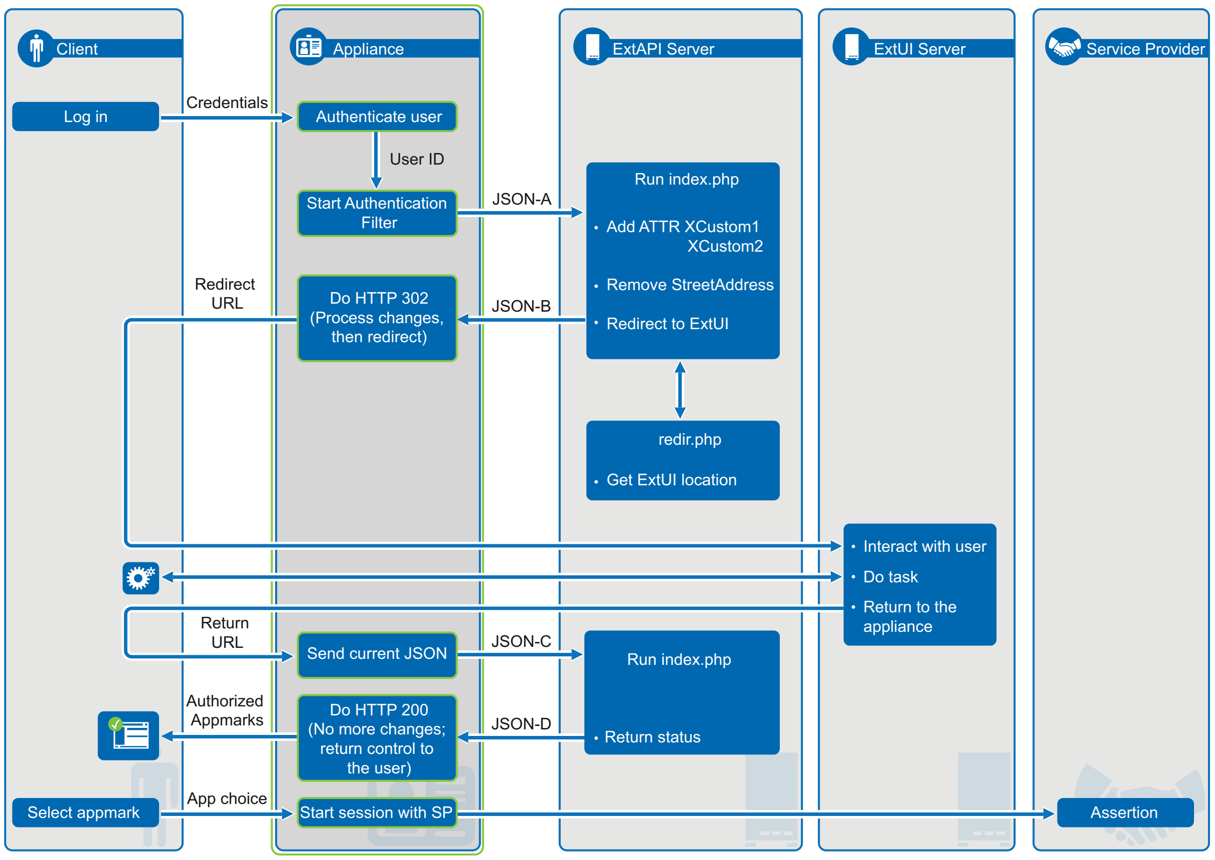
Example: Adding and Removing Identity Attributes with User Interaction

Your custom authentication scripts can require interaction with the user to gather input. The user is aware only of the actions that gather the user’s input. After the external authentication actions and associations complete successfully, the user can access the SaaS web service or application. CloudAccess establishes the session by sending the modified identity information.

The following figure shows how you can extend the authentication process to manipulate identity attributes for an authenticated user, and redirect control to another authentication script to perform business-specific logic that requires user interaction. For an example of how to add identity attributes, to set values for new or existing identity attributes, and to remove an identity attribute, see “Example:

ExtAPI Script" on page 11. For an example redirect script, see "Example: Redirect Script" on page 11.

Using the Authentication Filter to Add or Remove Identity Attributes with User Interaction



To prepare the Authentication Filter, do the following as an administrator user:

1. Create an authentication script, such as `index.php`, and store it on the ExtAPI server. For example:

   http://*extapi_server_dns*/extapi/index.php

2. Create a redirect script, such as `redir.php`, and store it on the ExtAPI server. For example:

   http://*extapi_server_dns*/extapi/redir.php

3. Create a user interaction script, such as `user-input.php` and store it on the ExtUI server. For example:

   http://*extui_server_dns*/extui/user-input.php

4. Enable the Authentication Filter tool in CloudAccess, and configure it to point to the target authentication script.

The extended authentication process is as follows:

1. The user logs in to the CloudAccess login page with the single sign-on login credentials.
2. CloudAccess authenticates the user credentials against the identity source.
3. The Authentication Filter intercepts the authentication process and sends a JSON POST that contains the user's identity information to the URL of a custom authentication script on the ExtAPI server.

4. The script sets values for the attributes XCustom1 and XCustom2, and removes the attribute StreetAddress.
5. The authentication logic determines that an action requires user interaction, and redirects the user's browser to a specified URL on the ExtUI server.
6. The target ExtUI script uses ExtUI library commands to perform the required business-specific logic to get information from the user, then returns to CloudAccess.
7. CloudAccess sends the current identity information with a JSON POST to the ExtAPI server to ensure that all encoded associations are complete.
8. After all external authentication is complete, the authentication script returns a JSON response to CloudAccess that contains an HTTP return status and the user's modified identity attributes for the session.
9. CloudAccess stores the modified identity information in memory; it does not store it in the user's identity source.
10. CloudAccess completes the authentication process and returns control to the user's web browser.
11. For the life of this session, the modified identity attributes are available to service providers when the user accesses a SaaS web service or application.

Example: JSON Request

In the following sample JSON request (JSON-A), the identity attributes specify the information for the authenticated user. Only the specified attributes can be manipulated by the ExtAPI. All other values are static information based on the session:

- The target file URL
- The DNS name of the appliance
- The user's web browser
- The user's ID GUID
- The return URL for the user's session

```
Request destination URL: http://extapi_server_dns/extapi/index.php

POSTed Data:{
        "API":{      "version":"0"     },
        "Request":{
                    "Host":"appliance_dns",
                    "User-Agent":"Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/33.0.1750.146 Safari/537.36"
                  },
        "Session":{
                     "ID":"448484815b1a417abf07b0a64d75961e-D5A1E4F8B4A0A1BD",
                     "ReturnURL":"https://appliance_dns/osp/a/t1/auth/app/
login?acAuthCardId=user&sid=0"
                  },
        "Identity":{
                "Principal-ID":"bis_EDIR_s1Vq9us-6a5828947d876047a4b86a5828947d87",
                "Attributes":{
                        "UserName":"admin",
                        "IdentityType":"EDIR",
                        "LastName":"admin",
                        "BirthDate":"Unavailable",
                        "Email":"admin@acme-widgets.com",
                        "FullName":"Admin admin",
                        "Gender":"Unavailable",
                        "ID":"6A5828947D876047A4B86A5828947D87",
                        "FirstName":"Admin",
                  }
        }
}
```

Example: JSON Response

The JSON response can modify only the identity attributes. The following code sample shows the portion of the JSON-B response that sets the values for XCustom1 and XCustom2. It sets XCustom1 to `value`. It sets XCustom2 to two values: `value2a` and `value2b`.

```
{"Identity":
  {"Attributes":
     {"set":{
      "XCustom1":"value",
      "XCustom2":["value2a","value2b"]}
}}}
```

In addition to setting or removing existing attributes, the response should contain a valid HTTP Response code. If the response returns an HTTP 200 OK response value, the appliance processes the specified changes, and then returns with a successful authentication to grant the user access to the authorized appmarks.

Example: ExtAPI Script

The following `index.php` script shows one way to use the ExtAPI library commands to perform the authentication logic that is described in "Example: Setting Values and Removing Identity Attributes" on page 7 and "Example: Adding and Removing Identity Attributes with User Interaction" on page 8.

```
<?php
    $retVal = "";
    $redirectLocation = "http://extapi_server_dns/extapi/redir.php";
       header('Content-type: application/json', true, 200);
    $rawJSON = file_get_contents("php://input");
        error_log("raw input: " . $rawJSON);
    $inVal = json_decode($rawJSON, TRUE);
    if ( !isset( $inVal['Identity']['Attributes']['XCustom1'])) {
        error_log("setting XCustom1 and XCustom2, redir to: ".$inVal['Session']['ReturnURL']);
        $retVal = json_encode(array("Identity" => array("Attributes" => array("set" =>
array("XCustom1" => "value", "XCustom2" => array("value2a", "value2b"), "XCustom3" => "1")))));
        header('Location: '.$redirectLocation.'?sendTo='.
urlencode($inVal['Session']['ReturnURL']), true, 302);
    } else if ( !isset( $inVal['Identity']['Attributes']['XCustom4'])) {
        error_log("setting XCustom4 and XCustom5, removing xCustom2");
        $retVal = json_encode(array("Identity" => array("Attributes" => array("set" =>
array("XCustom4" => "value4", "XCustom5" => "true"), "remove" => "XCustom2"))));
    }

     echo $retVal;
     error_log("returning: " . $retVal);
    die();
?>
```

Example: Redirect Script

The following `redir.php` script shows one way to specify the redirect control to the ExtUI script in "Example: Adding and Removing Identity Attributes with User Interaction" on page 8.

```
<?php
    error_log("redirecting to: ".$_GET['sendTo']);
  header('Location: '.$_GET['sendTo'], TRUE, 302);
?>
```