eRaider Web Sign-in API Reference, v08.1 Rev 2  
May 4th, 2009

This document describes the application programming interface (API) used to validate credentials provided to web applications during the eRaider Web Sign-in process (eRWS). These stored procedures are implemented on a Microsoft SQL Server. Details regarding the connection information, execute rights, and other database-specific details, are provided during the process of registering your application’s URL.

The credentials provided during eRWS are the web client’s IP address, eRaider username, and a temporary “login key.” These credentials are provided as parameters in the URL query-string of the web application’s registered application URL.

The validation of eRWS credentials need only be performed once per application session per user. Once the eRWS credentials have been validated for a specific user in a specific application, session state information can maintain the authenticity for the duration of their application session without the need to repeatedly execute the stored procedures to validate the credentials.

Deprecated Procedures

The following table lists stored procedures which should no longer be used in applications:

|  |  |
| --- | --- |
| Deprecated Procedure | Replace with… |
| eRaiderIntLookupBySSN | *No replacement* |
| eRaiderIntLookupByUsername | eRaiderLookupByUsername\_v081 |
| eRaiderLoginCheckKey | eRaiderLoginCheckKey\_v081 |

Authentication Procedure

Procedure:

eRaiderLoginCheckKey\_v081 @username, @login\_key, @ip

Parameters:

|  |  |  |
| --- | --- | --- |
| @username | varchar(255) | The username asserted to the eRaider-enabled application by the user’s browser. |
| @login\_key | varchar(255) | The temporary login key asserted to the eRaider-enabled application by the user’s browser. |
| @ip | varchar(255) | The IP address asserted to the eRaider-enabled application by the user’s browser. |

Returns Single-row Record-set:

|  |  |  |
| --- | --- | --- |
| valid | integer | 1 if credentials are valid. 0 if not. |
| fname | varchar(255) | If (valid=1) contains the legal first name of the authenticated user, otherwise NULL. |
| nname | varchar(255) | If (valid=1) contains the nick-name of the authenticated user, otherwise NULL. |
| mname | varchar(255) | If (valid=1) contains the legal middle name of the authenticated user, otherwise NULL. |
| lname | varchar(255) | If (valid=1) contains the legal last name of the authenticated user, otherwise NULL. |
| jobtitle | varchar(255) | If (valid=1) contains the jobtitle of the authenticated user, otherwise NULL. |
| addr1 | varchar(255) | If (valid=1) contains the first street address line of the authenticated user, otherwise NULL. |
| addr2 | varchar(255) | If (valid=1) contains the second street address line of the authenticated user, otherwise NULL. |
| city | varchar(255) | if (valid=1) contains the city portion of the postal address for the authenticated user, otherwise NULL. |
| state | varchar(255) | if (valid=1) contains the state portion of the postal address for the authenticated user, otherwise NULL. |
| zip | varchar(255) | if (valid=1) contains the zip/postal code portion of the postal address for the authenticated user, otherwise NULL. |
| phone | varchar(255) | if (valid=1) contains the phone number of the authenticated user, otherwise NULL. |
| email | varchar(255) | If (valid=1) contains the e-mail address of the authenticated user, otherwise NULL. |
| eRaiderID | char(10) | If (valid=1) contains the eRaider ID number of the authenticated user, otherwise NULL. |
| bannerID | varchar(255) | If (valid=1) contains the Banner ID of the authenticated user, otherwise NULL. |

User Look-up Procedures

User look-up procedures make it easy to get meaningful user information given some user-identifier.

*NOTE: All user look-up procedures make use of output parameters to return data, and also return a single-row record-set of the following format:*

|  |  |  |
| --- | --- | --- |
| username | varchar(255) | eRaider username |
| fname | varchar(255) | legal first name |
| nname | varchar(255) | nick-name |
| mname | varchar(255) | legal middle name |
| lname | varchar(255) | legal last name |
| email | varchar(255) | e-mail address |
| disp\_fname | varchar(255) | first name portion of display name |
| disp\_lname | varchar(255) | last name portion of display name |
| eRaiderID | char(10) | eRaider ID number |
| bannerID | varchar(255) | Banner ID |

Procedures:

eRaiderLookupByeRaiderID\_v081 //10-digit eRaider ID is known  
@eRaiderID char(10)  
@username varchar(255) output *(optional)*  
@fname varchar(255) output *(optional)*  
@nname varchar(255) output *(optional)*  
@mname varchar(255) output *(optional)*  
@lname varchar(255) output *(optional)*  
@email varchar(255) output *(optional)*  
@disp\_fname varchar(255) output *(optional)*  
@disp\_lname varchar(255) output *(optional)*  
@bannerID varchar(255) output *(optional)*

eRaiderLookupByUsername\_v081 //Username is known  
@username varchar(255)

@fname varchar(255) output *(optional)*  
@nname varchar(255) output *(optional)*  
@mname varchar(255) output *(optional)*  
@lname varchar(255) output *(optional)*  
@email varchar(255) output *(optional)*  
@disp\_fname varchar(255) output *(optional)*  
@disp\_lname varchar(255) output *(optional)*  
@eRaiderID char(10) output *(optional)*  
@bannerID varchar(255) output

eRaiderLookupByBannerID\_v081 //Banner ID is known  
@bannerID varchar(255)

@username varchar(255) output *(optional)*  
@fname varchar(255) output *(optional)*  
@nname varchar(255) output *(optional)*  
@mname varchar(255) output *(optional)*  
@lname varchar(255) output *(optional)*  
@email varchar(255) output *(optional)*  
@disp\_fname varchar(255) output *(optional)*  
@disp\_lname varchar(255) output *(optional)*  
@eRaiderID char(10) output *(optional)*

**Appendix A:** Programmatic Method of Windows Domain Controller Discovery  
*Added May 4th, 2009*

Applications which make use of eRaider Web Sign-in may have a need to obtain further data from the University’s Windows Active Directory system. In such cases, the following method should be used.

It is critical that specific domain controller names not be specified in applications. A dynamic method of discovery must always be used. For applications that do not require high availability or robustness, a “Simple Case” has been developed. For applications that require high availability and robustness, an “Advanced Case” has been developed. Both employ the same basic dynamic method of discovery using DNS.

Simple Case:

1. When specifying the name of the domain controller to which the application will connect, use the name of the respective Active Directory domain (e.g., ttu.edu, or ttuhsc.edu). For example, if an application needs to connect to a domain controller within the ttu.edu domain, then the domain controller name should be specified as “ttu.edu”.
2. Specifying the domain name in this manner will result in connecting to different domain controllers over time because the DNS system provides the IP addresses of each domain controller in a “round-robin” fashion.
3. It is possible that the IP addresses returned from DNS may not be available for connections at the time the application attempts to connect. To handle these types of situations, see “Advanced Case” below.

Advanced Case:

1. An application must query DNS for the address (A) record(s) of the appropriate Active Directory domain (e.g., ttu.edu or ttuhsc.edu).
2. DNS will return a list of IP addresses associated with the domain controllers for the specified domain.
3. The application then attempts to discover a responsive domain controller by the following means
   1. In the order of the list returned by DNS, the application attempts to connect to each domain controller.
   2. If an attempted connection succeeds, then the application uses that domain controller for further operations and terminates the discovery process.
   3. If an attempted connection fails, then the application must use the next domain controller IP address in the list and begin the connection attempt process again.
   4. If all connection attempts fail, then the application must handle the exception of being unable to connect to any domain controller.