SecureKey
Configuration Guide

Version 11
Last Updated: March 2014

Logi ANALYTICS 11 Ad Hoc Reporting

Overview of Ad Hoc Security Models

Every Ad Hoc instance relies on a security model to determine the authentication process for a user and the authorizations granted to the user. Ultimately Ad Hoc must be able to determine the user, their role(s) and which organization the user is associated with regardless of the security model chosen.

Follow are brief descriptions of the various security models.

Standard – as the default security model, credentials supplied in a login page are verified by interrogating the Ad Hoc metadata database associated with the instance. Users, organizations and roles must have been created by the System Administrator and are managed by the System Administrator.

Database – similar to the Standard security model, the Database model relies upon a database external to Ad Hoc to verify the logon credentials. Users are managed outside of Ad Hoc. Organizations and roles are managed within Ad Hoc.

Session – the user is authenticated elsewhere, typically a parent application, and the user name is passed to Ad Hoc. Roles and organizations are managed within Ad Hoc.

NT Authentication – the user is authenticated by the Windows operating system. Roles and organizations are managed within Ad Hoc. User/Role associations are adjusted by comparing the User/NT User Group to the User/Role(s) in the Ad Hoc metadata. The is one of the single sign-on security models.

SecureKey – the user is authenticated by a parent application. Access to Ad Hoc is controlled by determining if the IP address of the caller and the “approved” IP addresses match or grant access. Roles and organizations must be managed within Ad Hoc. User/Roles are adjusted to match the information provided by the parent application. This is one of the single sign-on security models.

Configuring SecureKey Authentication

The Logi SecureKey authentication method is used to move the authentication of the user outside of Ad Hoc and into a “parent” application. It provides improved integration for applications requiring secure access to a Logi application. It supports the "single sign-on" model while enhancing security because user credentials do not need to be exposed in a query string or otherwise passed to Ad Hoc in order to authenticate the user.

With SecureKey, users are authenticated in a parent application. When the parent application requires access to an Ad Hoc application, a request is made to acquire a key from background service and exchanged between the two applications. Once established, the key can be used only once but the authentication remains valid for the life of the session.

This section provides assistance for systems administrators who want to authenticate users with the Logi SecureKey methodology. There are basically two steps required to configure the application to authenticate via SecureKey:

1. Configure the application’s *_Settings.lgx* file for SecureKey authentication
2. Code the SecureKey processing in the parent application

Configure the _Settings.lgx file

To configure the application to use SecureKey authentication:

1. Make a backup of the */_Definitions/_Settings.lgx* file
2. Open the file in a text editor (eg. Notepad)
3. Locate the <Security> element
4. Replace the <Security> element with the one provided below
5. Revise the value(s) for the AuthenticationClientAddresses attribute to the appropriate IP addresses. This value can be determined by "pinging" the application server from the web server hosting your Ad Hoc application. If one computer serves both purposes and Ad Hoc is being called using *localhost* in the URL, enter 127.0.0.1 for this value. Multiple IP addresses must be separated by commas.
6. **Save** the file

```
  <Security SecurityEnabled="True" AuthenticationSource="SecureKey"
CacheRights="Session" AuthenticationClientAddresses="10.10.10.1" />
```

SecurityEnabled. Enables or disables security.

AuthenticationSource. Sets how the server determines the current user. "SecureKey" makes it possible for another application to do the authentication, then pass the user to this application. The other "main" application calls rdTemplate/rdGetSecureKey.aspx, passing Username (and possibly Roles and other session information), then getting back a key. Then the main application redirects the user to this application passing the key as a parameter rdSecureKey. If Roles are passed, the user/role relationships will be managed within Ad Hoc.

CacheRights. When set to "Session", Roles and Rights are only determined at the beginning of the session. Otherwise, Roles and Rights are determined with each request. “Session” is required when the AuthenticationSource is set to “SecureKey”.

AuthenticationClientAddresses. Used only when AuthenticationSource="SecureKey", AuthenticationClientAddresses is a comma-delimited list of IP addresses for approved external applications that will be requesting keys under Logi SecureKey Authentication. AuthenticationClientAddresses is required for AuthenticationSource="SecureKey". If rdGetSecureKey.aspx is called with the "localhost" address, enter "127.0.0.1".

Configure the Parent Application

The parent application is responsible for authenticating the user, establishing the link to Ad Hoc, providing the information to properly set the requisite session parameters, and controlling the request redirects.

Ad Hoc must ultimately be able to determine the user, roles, and organization for a user. For existing users, all that is necessary to determine the roles and organization is the user name. For new users, either the user information (user name, roles, organization) must be managed via the Ad Hoc interface (or through an application using the API) or the parent application can marshal the information and have it available to Ad Hoc via session parameters. In the latter case, Ad Hoc will create the user and associate the user with the provided organization and roles automatically.

The parent application must perform two actions; 1) acquire a valid key and 2) launch the Ad Hoc application.

Acquire a Valid Key

When the user first requests a report or web page from the Ad Hoc, the parent application issues a request to the Ad Hoc's AuthenticationService using this format:

```
http://myServer/myAdHoc/rdTemplate/rdGetSecureKey.aspx?Username=bob
&ClientBrowserAddress=10.10.10.1
```

The AuthenticationService (rdGetSecureKey) returns a secure key to the parent application. In addition, the service also sets session information based on the parameters passed via the URL.

In the above example, a session variable is created for the user name and set to the value of “bob”. A session variable for the anticipated client browser address is also created and set to “10.10.10.1”.

If users are managed wholly with Ad Hoc, the above example is sufficient information to potentially grant access to Ad Hoc.

If users are to be automatically added to the Ad Hoc metadata database and have their roles and organization associated with the user name, the request to rdSecureKey must also pass the role and organization information. The following format:

```
http://myServer/myAdHoc/rdTemplate/rdGetSecureKey.aspx?Username=bob
&Roles=”End User”&ahUserGroupID=1&ClientBrowserAddress=10.10.10.1
```

In the above example, the request for a key process will 1) set the user name session variable to “bob”, 2) the roles session information to be “End User”, and 3) the ahUserGroupID to be 1.

In this model the assigned roles must exist within Logi Ad Hoc.

Launch the Ad Hoc Application

The AuthenticationService returns a secure key to the parent application, which then redirects the user's request back to Ad Hoc’s URL, passing the secure key as REQUEST or POST parameter. For example,

```
http://myServer/myAdHoc/Gateway.aspx?rdSecureKey=ABC123456789
```

Ad Hoc then verifies the incoming key to make sure that the request has been initiated by a valid client and verifies that the user, roles and organization can be determined and lets the user in.

If the user, roles and organization have been created as session information, Ad Hoc will either create the user using the information or adjust the existing users’ roles according to the session information provided.

For additional details and a coding example, visit SecureKey on DevNet and click the Configuring a Parent Application link.

Additional Configuration Notes

Compatibility With Prior Versions

In the configuration outlined above, the <Security> element identifies the security method and the security process assumes that the three elements (user, organization, and roles) that Ad Hoc needs to work properly will be passed as session information. After Version 10.1, no other elements are required to implement this security model. In versions of Ad Hoc prior to 10.1, one additional element is required to identify the source of the user's roles:

```
<UserRoles Type="SecureKey" ID="SomeID" />
```

This element is a child of the <Security> element, so the general structure of the <Security> element would be:

```
<Security … >
     <UserRoles … />
</Security>
```

Using Existing Roles

In the configuration above where the user name, organization, and roles are being passed to Ad Hoc, the expectation is that Ad Hoc will automatically manage the user information in the metadata. If the user does not exist in the Ad Hoc metadata database, they will be added and assigned to the organization and the designated roles will be associated with them. If the user exists in the metadata, the roles will be verified and adjusted to match the incoming role information.

There are configuration models where the user information is manually created and maintained via the Ad Hoc interface. In this scenario the SecureKey security method is not expected to manage the user information, however, the authorization for the user still must be determined. Since authorization is controlled by a user's assigned roles, the security model must determine those roles by examining the metadata database.

To accomplish that goal, a <UserRoles> element must be added as a child of the <Security> element.:

```
<Security … >
      <UserRoles Type="SQL" ID="UserRole" Source="SELECT RoleID FROM
UserRole, Users WHERE Users.UserName='@Function.UserName~' AND
UserRole.UserID = Users.UserID" ConnectionID="ahMetadata" />
</Security>
```

This element is identical to the <UserRoles> element found in the typical, Standard <Security> element. This instructs the application to determine the user's roles by interrogating the metadata database based on the user name.

Obviously using this method the user information must exist in the metadata database before a user will be granted access. New users will not be automatically created nor will the user's roles be adjusted using this method.