Contents

We've covered the basics of authentication and authorization previously, but there are still **additional steps** you can take in building and deploying a secure application. We've broken these down into configuration-related opportunities - considerations which can be addressed through the deployment configuration; and coding-relating - additional choices you can make in your code.

Configuration

Use SSL to secure communications. The default DevForce configuration will automatically use a secure transport if you specify a URL beginning with https in the objectServer tag. If your Silverlight application was downloaded via a URL beginning with https, then communications to the EntityServer will use a secure transport. See here for more information on configuring SSL in IIS.

Disallow access to the debug log file. In IIS, unless you restrict access, anyone can browse to your debuglog.xml file and discover much more about your application than you probably want to share. To see if your debug log is browsable, just point your internet browser to it - for example http://localhost/badgolf/log/debuglog.xml. Blocking access is easy and there are many ways to do it; here are a few options: 1) change the logFile path to a secure folder outside of the IIS application folder, 2) turn off all NTFS permissions to the log folder except as required by the application pool account to create and append to the file, or 3) disable all IIS authentications for the folder.

In a client application, you may want to turn off the debug log altogether if it's not possible to secure the folder.

Use a database account with appropriate privileges. The userid defined in the database connection string should not be an administrator or DBA account.

Protect the configuration file. You can encrypt configuration sections, such as the connectionStrings and ideablade.configuration sections, to limit unauthorized access. See here for more information. (To encrypt the ideablade.configuration section, make sure that the section definition contains the fully-qualified assembly name.)

Consider encrypting database connections. See here for more information on encrypted SQL Server connections.

Do not publish service metadata. By default metadata publishing is disabled for DevForce services. If you enabled it for any reason, be sure to turn it off when you deploy your application.

Disable anonymous access in IIS if only authenticated users may access your application.

Do not use a "wide open" client access policy. If you do not support cross domain access, remove the clientaccesspolicy.xml and/or crossdomain.xml files. If cross domain access is needed, lock down the access granted. Note that the sample file provided by DevForce is "wide open" and is intended as a starter sample only.

Coding

Send less detailed error messages to the client. Although during development you want detailed error messages to help in diagnosing problems, your client applications should not receive error messages from the server which might contain detailed or sensitive information. Implement a custom EntityServerErrorInterceptor to examine and optionally replace the exception to be sent to the client.

Do not store connection strings in a client config file. If the application is an n-tier application the connection information is unnecessary, as only the *EntityServer* requires connection information. In a 2-tier application, consider using a custom *DataSourceKeyResolver*, which although not completely secure because your code can be disassembled, may still be more secure than the .config file.

Encrypt sensitive files. If you save an *EntityCacheState* or other sensitive information to the file system or Isolated Storage, consider encrypting it. Here's a sample.

Consider logging for audit purposes.

Consider the entity model exposure. The client application can be easily disassembled to see the entire generated entity model. Do you really want that User entity exposed? If you are using custom authentication with your own user database, consider not including these tables in your entity model, or creating a separate server-only entity model just for authentication purposes.

Consider WCF security. Depending on the URL, DevForce will default to either non-secure communications or transport security with SSL. For additional security requirements you can configure both client and service security using configuration file settings, or by implementing custom ServiceHostEvents and ServiceProxyEvents classes.