**eDOCS DM - DM Extensions - Logging on using the DM Extensions API**

**Applies to**

* eDOCS DM 10.0, 5.2.1, 5.3, 5.3.1

**Summary**

With Document Management (eDOCS DM), using DM Extensions, how can you log on using the DM Extensions API?

Resolution

The communication that occurs between the DM Clients and DM Server are transaction-based and hence, stateless. The DM Server assigns a DST (DM Security Token) to client applications that successfully log on. The DST is required for all transactions that occur between the Client tier application and the DM Server. The DST is a unique 64-character string used to secure access to DM.  
  
Before using any other constructs from DM Extensions, a user must first log on to DM. Logging on to DM is facilitated using the Application object. The Application object loads the DM Objects and Services Provider (FsPlugin.dll) and provides mechanisms to authenticate the user and password. Once the class is successfully instantiated, a process is spawned called DM.exe. The log on process and future interactions with DM occur through the DM.exe process.  
The Application object provides both UI and UI-less interfaces to authenticate a user. As part of the process of successfully authenticating credentials with DM, a DM Security Token is returned. The Application object stores the DST in the memory of the DM.exe process  
The Application class is a singleton class, meaning only one instance of the DM.exe process runs on a machine. Subsequent attempts to create the Application object provide a handle back to the Application object.   
  
**Displaying a Logon Window:**  
  
When the Application object is created, the Logon window is not automatically displayed. The Logon window is only displayed when a property/method is invoked on the Application object or an object is utilized from the DM Extensions API.   
The DM Extensions API uses DM API constructs to invoke operations with the DM Server. Since the communication between the Client tier applications and DM Server are transaction-based, a DST is required as part of each transaction. Unlike the DM API, the DM Extensions API does not require that the DST be explicitly utilized for each operation. The Application object maintains the DST in memory so that other DM Extensions API objects can invoke DMS functions with the DM Server without managing the DST.  
  
**Logging On to DM without UI:**  
It is possible to log on to DM without displaying the default Logon window to the user. Instead, a custom form could be used to retrieve the information and the process of logging on to DM can occur in the background. This functionality is useful in a custom application where a user may need to enter credentials.  
In order to achieve this, the username and user's password must be passed to the Application object using the Login method. In order to determine if the log on attempt was successful, the ErrCode property should be verified. If the value returned does not equal a value of OM\_SUCCESS (0), the log on attempt failed. Alternatively, the value from the Login method can be checked. If the value is a null string, the user did not successfully log on to DM. The reason for the logon failure can be displayed using the DisplayMessage method from the Application object. with DM. All DMOS API interfaces inherit from the IOMErrInfo interface, therefore, the properties ErrCode and ErrText are included as properties of the Application object.A list of values from the OMErrorCodes enumeration is provided at the end of this document   
If the logon attempt failed, any use of objects/properties/methods from the DM Extensions API will cause the Logon window to be displayed to the user.   
  
public void DocsObjectLoginNOUI(string Library, string User, string Password)

{  
  
string DST= string.Empty;  
Hummingbird.DM.Extensions.Interop.DOCSObjects.Application dmApp = new Hummingbird.DM.Extensions.Interop.DOCSObjects.Application();  
  
try  
{  
DST= dmApp.Login(Library, User, Password);  
  
if (dmApp.ErrCode != (int)OMErrorCodes.OM\_SUCCESS)  
{  
dmApp.DisplayMessage(dmApp.ErrText, (int)DisplayMessageTypes.DMT\_ERROR);  
  
//The next line will force a logon window to display if login failed because a property  
// of the Application object has been invoked   
DST = dmApp.DST;  
}  
else  
{  
dmApp.DisplayMessage("Logged into " + Library, (int)DisplayMessageTypes.DMT\_INFO);  
}   
}  
  
catch (Exception ex)  
{  
MessageBox.Show(ex.ToString());  
}  
  
  
}

**Auto Login to DM:**  
  
The DM Extensions API does not provide methods or properties to enable the automatic log on of a user into DM. This capability is provided by enabling the Allow Auto Logon option for a DM Library. The Allow Auto Logon option is enabled from Library Maintenance under System Parameters or Group Parameters. By default, when DM is installed, the system logon feature Allow Auto Logon is disabled.  
If the Allow Auto Logon capability is enabled, when the Application object is created, and it will obtain the login credentials, and the user will be logged into DM without displaying a login window and theApplication.Login() method is not needed.   
  
  
The BCSSession Object:  
  
The BCSession object is used to register your application with the DM.exe process. All clients that wish to interoperate with the DM.exe process must create a BCSession object and call either the \*1Logon method or call the LogonEx method. This is required in order to register \*2external applications with the DM.exe process -otherwise the DM process will unload inadvertently. When this occurs, any subsequent attempts to use Docsobjects methods or properties will fail, resulting in RPC errors.  
Important! - Once again, the BCSession object MUST be used to logon to DM extensions if your application is running outside of the DM.exe process.   
Upon close of the custom application, or when the custom application is done working with DM Extensions, the Logoff method of the BCSession object (BCSession.Logoff()) must be called. This method unregisters your application with the DM.exe process. Failure to call the Logoff method will result in the DM.exe process staying in memory. When this occurs, the only way to exit the DM.exe process is to either End Task on the process or to reboot the machine.  
Important! - Once again, the BCSession object MUST be used to logoff of DM Extensions if your application is running outside of the DM.exe process.

public void DocsObjectLoginNOUIWithBCS(string Library, string User, string Password)  
{  
  
string DST= string.Empty;  
Hummingbird.DM.Extensions.Interop.DOCSObjects.Application dmApp = new Hummingbird.DM.Extensions.Interop.DOCSObjects.Application();  
BCSession objBCSession = new BCSession();  
  
try  
{  
DST=dmApp.Login(Library, User, Password);  
  
if (dmApp.ErrCode != (int)OMErrorCodes.OM\_SUCCESS)  
{  
dmApp.DisplayMessage(dmApp.ErrText, (int)DisplayMessageTypes.DMT\_ERROR);  
  
//The next line will force a Logon Window to display if the login failed.  
DST = dmApp.DST;  
///  
  
if(!string.IsNullOrEmpty(DST))  
{  
objBCSession.LogonEx();  
}  
}  
else  
{  
dmApp.DisplayMessage("Logged into " + Library, (int)DisplayMessageTypes.DMT\_INFO);  
objBCSession.LogonEx();  
}   
}  
  
catch (Exception ex)  
{  
MessageBox.Show(ex.ToString());  
}  
  
finally  
{  
  
objBCSession.LogOff();  
  
}  
  
}

\*1 To call the Logon method of the BCSession object (BCSession.Logon()), your application must implement the BCBrowser interface. It is more common for custom applications to call the LogonEx method of the BCSession object (BCSession.LogonEx()) as the BCBrowser interface is implemented automatically for you.  
\*2 External applications refer to applications that do not implement the IOMEventHandler interface

Additional Information

Application Object Methods and Properties Used:  
  
string Login(string bstrLibrary, string bstrUserName, string bstrPassword)  
- bstrLibrary: The name of the DM Library to log the user in to.  
- bstrUserName: The user name for the specified user.  
- bstrPassword: The password for the specified user.  
  
  
void DisplayMessage(string bstrMessage, int nType)  
- bstrMessage: The message to display to the user.  
- nType: The type of message. This must be one of the values of the DisplayMessageTypes enumeration  
  
  
DisplayMessageTypes Enumeration:   
DMT\_NONE Simple Message  
DMT\_ERROR Error Message  
DMT\_INFO Info Message  
DMT\_WARNING Warning Message  
  
The IOMerrInfo Interface  
  
Almost all of the DM Extensions API objects inherit from the IOMerrorInfo class. This provides common properties used to retrieve error information.   
IOMerrorInfo class properties are as follows:  
  
int ErrCode { get; }   
  
This property returns a numeric error code from the last method executed on an object. The error codes returned are one of the values from the OMErrorCodes enumeration. The following table lists possible error codes that can be returned.  
  
string ErrText { get; }   
  
This property returns a description of the error from the last method executed on an object as a string. An empty string indicates that no error has occurred.  
  
  
OMErrorCodes Enumeration Description:  
  
OM\_SUCCESS Indicates that an operation succeeded.  
OM\_CANCELLED\_BY\_USER Indicates that an operation was cancelled by the user.  
OM\_SYSTEM\_ERROR Indicates that a system error occurred.  
OM\_INVALID\_ARGUMENTS Indicates that invalid arguments were specified for a method.  
OM\_INSUFFICIENT\_RIGHTS Indicates that the user has insufficient  
rights to complete an operation  
OM\_ERROR\_LOADING\_MODULE Indicates that an error occurred when loading an external module.  
OM\_ERROR\_LOAD\_FORM Indicates that an error occurred when loading a form.  
OM\_ERROR\_IN\_EXTERNAL\_  
MODULE Indicates that an error in an external  
module occurred  
OM\_INVALID\_OBJECT Indicates that a method cannot be invoked for an object   
OM\_INVALID\_OBJECT\_STATE Indicates that an object is an in an invalid state.  
OM\_FUSION\_ERROR Indicates a DM Server Error has occurred.  
OM\_INTERNAL\_ERROR Indicates an internal error has occurred.  
OM\_UNPLUGGED\_ERROR Indicates the unplugged mode does not support this property or method.