Aras IOM SDK

Programmer's Guide

Document #: D-009381

Last Modified: *04/30/2025*

Copyright Information

Aras Corporation
100 Brickstone Square
Suite 100
Andover, MA 01810

Notice of Rights

Notice of Liability

THIS DOCUMENT IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY, AND THE CONTENTS HEREOF ARE SUBJECT TO CHANGE WITHOUT NOTICE. THE INFORMATION CONTAINED IN THIS DOCUMENT IS DISTRIBUTED ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE OR A WARRANTY OF NON-INFRINGEMENT. ARAS SHALL HAVE NO LIABILITY TO ANY PERSON OR ENTITY WITH RESPECT TO ANY LOSS OR DAMAGE CAUSED OR ALLEGED TO BE CAUSED DIRECTLY OR INDIRECTLY BY THE INFORMATION CONTAINED IN THIS DOCUMENT OR BY THE SOFTWARE OR HARDWARE PRODUCTS DESCRIBED HEREIN.

Table of Contents

Introduction

This document is a comprehensive guide for developers working with the Aras Innovator Object Model (IOM). Whether building integrations or extending the Aras platform through custom development, this guide will help leverage the IOM library's capabilities effectively.

The Aras IOM library provides a robust and flexible object-oriented API for interacting with Aras Innovator. Using an intuitive, object-based approach, it enables developers to construct and submit AML (Aras Markup Language) queries to the Innovator server.

This guide covers the essential aspects of programming with the IOM, including:

- Installation and setup
- Establishing connections and authentication
- Understanding and using the IOM API and its methods

It is intended to serve both as a developer reference and a practical user guide, featuring:

- A detailed overview of the IOM (Innovator Object Model), the object-based API for AML
- A cookbook of common programming tasks and patterns

Whether using the IOM library within the Aras environment (e.g., in server methods or other extensibility points) or from an external application to integrate with Aras Innovator, this guide provides the foundational knowledge and best practices needed to get started and succeed.

1 IOM (.NET Framework / .NET)

This section describes using the Aras Innovator Object Model (IOM) in .NET Framework and .NET Core/.NET 6+ environments.

1.1 Installation and Prerequisites

The Aras Innovator Object Model (IOM) is available in multiple formats to support a variety of development environments and technologies. This section describes locating, installing, and referencing the IOM library for different use cases.

Variant	Target Environment
IOM for .NET Framework	.NET Framework 4.7.2+
IOM for .NET	.NET 6, .NET 8

1.1.1 IOM for .NET Framework

The classic .NET Framework version of the IOM library is distributed as a DLL and is located in:
A .NET version of IOM.dll can be found in the `.NET` folder within the `Aras IOM SDK\<Version>\IOM-<Version>-SDK.zip` archive on the FTP.

To install and use:

1. Copy IOM.dll to a suitable location in the project directory.
2. In Visual Studio project, right-click References > Add Reference...
3. Browse to and add IOM.dll.
4. Ensure the project targets .NET Framework 4.7.2 or higher.

1.1.2 IOM for .NET

A NuGet package version of the IOM library is provided for use with modern .NET versions.
Packages are published to nuget.org under official Aras feed. Users can also find in the `Packages` folder within the `Aras IOM SDK\<Version>\IOM-<Version>-SDK.zip` archive on the FTP.

Recommended Approach: Installing from nuget.org

1. Open Visual Studio and load .NET project.
2. Right-click the project in Solution Explorer and choose Manage NuGet Packages...
3. Ensure nuget.org is selected under Package source.
4. Search for the IOM package (e.g., Aras.IOM) by name.
5. Install the desired version.

Note: For detailed instructions on configuring and using NuGet packages in Visual Studio, see the official documentation at https://learn.microsoft.com/en-us/nuget/consume-packages/install-use-packages-visual-studio

This approach leverages the public NuGet ecosystem, making it simpler to stay up-to-date with any future IOM package releases and eliminating the need to manage a local feed.

Alternative: Local NuGet Repository:

If nuget.org is inaccessible or the environment requires an internal feed, you can set up a local NuGet repository. For detailed instructions, see Appendix A—Local NuGet Repository.

1.2 Basic Connection using Username/Password

Before working with Aras Innovator through the IOM library, every application must establish a connection to the Aras Innovator server and perform a login. Once authenticated, an instance of the Innovator class is created, which becomes the main interface for sending and receiving AML-based requests.

This section demonstrates how to establish a basic connection using a server URL, database name, username, and password.

> **Note:** For secure and scalable authentication using OAuth 2.0 access tokens, see Section – Authentication with Access Tokens (OAuth 2.0).

Basic Connection Flow

1. Create an HttpServerConnection using the server URL, database name, username, and password.
2. Call the Login() method on the connection.
3. If login is successful, use IomFactory.CreateInnovator() to create an Innovator instance.
4. Use the Innovator instance to interact with Aras Innovator via IOM.
5. When done, call Logout() on the connection to cleanly end the session.

> **Note:** When working inside Aras Innovator, such as in Server Methods or other platform extension points, you typically do not need to manually establish a connection or log in. In these cases, the platform manages the connection and authentication for you. The key step is to obtain the Innovator instance using the appropriate API for your context. In some environments, you may use *this.getInnovator()*. In others, you might be given a connection and need to call *IomFactory.CreateInnovator(connection)*. The exact method depends on where your code is running, refer to the appropriate documentation for that specific extension point.

Sample Code

```
using Aras.IOM;
…
string url = "https://myserver/MyInnovator/Server/InnovatorServer.aspx";

// These values should be obtained from user input or configuration
string db = "Innovator";       // Database name
string userName = "admin";     // Username
string password = "password";  // Plaintext password
```

```
HttpServerConnection conn = IomFactory.CreateHttpServerConnection(url, db,
userName, password);
Item loginResult = conn.Login();

if (loginResult.isError())
{
   throw new Exception("Login failed: " + loginResult.getErrorString());
}

Innovator innovator = IomFactory.CreateInnovator(conn);

// Use 'innovator' to perform queries...
…
```

Logging Out

It is strongly recommended to explicitly log out of Aras Innovator when the application is finished.

```
…
conn.Logout();
…
```

This ensures the session is properly closed and server resources are released.

1.3 Authentication with Access Tokens (OAuth 2.0)

Modern applications often require secure, token-based authentication to comply with enterprise security standards, support multi-system integrations, or enable single sign-on (SSO). Aras Innovator supports OAuth 2.0 authentication flows through the IOM library, enabling developers to obtain and use access tokens in place of direct user credentials.

This section covers supported authentication flows, how to discover the OAuth server configuration, and how to use access tokens with the IOM API.

1.3.1 Overview

The IOM API for authentication supports multiple authentication grants to obtain access token: a Resource Owner Password Credentials grant (further on Password grant), an Authorization Code grant and our custom Impersonate grant (a custom grant based on Client Credentials grant). Users can use an obtained access token in subsequent requests to the Aras Innovator server.

Note: See more about OAuth 2.0 authorization grants in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4. Refer to the Impersonate grant chapter for more information.

The IOM API for authentication includes a public *ITokenProvider* interface and its default implementations: *PasswordTokenProvider, ImpersonateTokenProvider, AuthorizationFlowTokenProvider* and *WindowsTokenProvider*. If the default implementation does not cover your needs, it is possible to provide a custom implementation of the *ITokenProvider*.

Warning Before creating a token provider, it is necessary to get a *DiscoveryDocument* that contains information about OAuth server configuration.

1.3.2 Discovery of OAuth Server

The Discovery mechanism allows clients to find the OAuth server and request its configuration.

The IOM OAuth API includes a public *IDiscoveryDocumentProvider* interface and its default implementation of the *DiscoveryDocumentProvider*.

DiscoveryDocumentProvider does the following:

1. Sends a request to http://<company.net>/<Innovator>/Server/OAuthServerDiscovery.aspx to determine the OAuth server location. OAuth server URLs are specified in InnovatorServerConfig.xml and look like the following:

```
<OAuthServerDiscovery>
    <Urls>
        <Url value="http://company.net/Innovator/OAuthServer/" />
        <!-- Url value="http://company.local/Innovator/OAuthServer/" /-->
    </Urls>
</OAuthServerDiscovery>
```

1. Requests the OAuth server discovery configuration.

The *DiscoveryDocumentProvider* receives all OAuth server URLs that are configured in *InnovatorServerConfig.xml* and prioritizes the list of OAuth server URLs according to the following order:

1 Last accessible OAuth server URL if presented.

2 URLs with the same host as the Aras Innovator server host.

3 All other URLs received from *OAuthServerDiscovery.aspx*.

After receiving the OAuth server URLs, the *DiscoveryDocumentProvider* tries to find an accessible path by requesting */.well-known/openid-configuration* paths from the list according to priority. After an accessible path is found, *DiscoveryDocumentProvider* creates an instance of *DiscoveryDocument* based on the response from the OAuth server and saves the last accessible URL.

The DiscoveryDocument contains information about:

- OAuth server endpoints (e.g., authorization and token endpoints),
- Protocol type (standard or custom),
- Protocol version.

This information is used while creating token providers in cases where it is necessary to provide the *DiscoveryDocument* by itself or some data from it.

The following example shows how to use the *DiscoveryDocumentProvider*:

```
var discoveryDocumentProvider = new DiscoveryDocumentProvider(
   "https://myserver/MyInnovator/Server/InnovatorServer.aspx");

DiscoveryDocument discoveryDocument =
   await discoveryDocumentProvider.GetDiscoveryDocumentAsync();
```

Note: The GetDiscoveryDocumentAsync method can take CancellationToken as a parameter to provide the possibility of canceling a task.
The default DiscoveryDocumentProvider implementation does not use a cache for DiscoverDocument. If caching is necessary for your application, it is possible to implement a wrapper with the cache.

The *DiscoveryDocument* instance has the following properties:

- `Issuer` – token issuer identifier (OAuth server).
- `JwksUri` – JSON Web Key Set document URI.
- `AuthorizeEndpoint` – URL of authorization endpoint.
- `TokenEndpoint` – URL to token endpoint.
- `EndSessionEndpoint` – URL of end session endpoint.
- `ProtocolVersion` – version of protocol that is used by the OAuth server.
- `ProtocolInfo` – information about protocol that is used by the OAuth server.
- `ProtocolType` – type of protocol that is used by the OAuth server.
- `KeySetJson` – JSON web key set.

A Protocol is used to determine what authentication header is used by the OAuth server. The *Standard* protocol type uses the `"Authorization"` header and 401 HTTP status code for `"Unauthorized"`, the *Custom* protocol type uses `"X-Aras-Authorization"` and 498 HTTP status code for `"Unauthorized"`. The *Custom* protocol type is necessary in case the *Standard* protocol type conflicts with other protection mechanisms (e.g., when all server resources are protected by Windows authentication).

The *DiscoveryDocument* class also makes it possible to get any other property from a discovery document with the following methods:

- `TryGetString(string name)` – gets a string value from the discovery document by name. If the property is not found, it returns *null*.
- `TryGetBoolean(string name)` – gets a boolean value from the discovery document by name. If the property is not found or is not of the boolean type, it returns *false*.
- `TryGetStringArray(string name)` – gets a string array value from the discovery document by name. If the property is not found or is not of the string array type, it returns *null*.

Note: For more information about discovery see https://openid.net/specs/openid-connect-discovery-1_0.html.

1.3.3 Password Grant

The Password grant is useful in cases where the client can obtain the user credentials. Password grants are often used for legacy or migration reasons. It is recommended to use other grant types whenever possible.

Note: See more about Password grants in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4.3

Users must use *the PasswordTokenProvider* to get an access token using a Password grant... Users must pass *PasswordTokenProviderOptions* to create the *PasswordTokenProvider:*

```
// database, username and password values should be entered by user
String db;
String userName;
String password;

var tokenProvider = new PasswordTokenProvider(
   new PasswordTokenProviderOptions()
   {
      TokenEndpoint = discoveryDocument.TokenEndpoint,
      ClientId = "IOMApp",
      Scope = "Innovator",
      Database = db,
      UserName = userName,
      Password = password
   });
```

PasswordTokenProviderOptions use the following options:

- `TokenEndpoint` – URL of OAuth server token endpoint.
- `ClientId` – ID of OAuth client that requests token.
- `Scope` – scope that will be used to request token. Currently all Aras Innovator resources require *Innovator* scope.
- `Database` – name of database to login to.
- `UserName` – user name.
- `Password` – user password.

Note: A password can be passed either as plain text or MD5/SHA256 hash (depending on FIPS mode).

To get an access token it is necessary to call the *GetAccesTokenAsync* method. This method sends a request to the OAuth server token endpoint and returns the access token created by a Password grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

Note: The GetAccessTokenAsync method can take CancellationToken as a parameter to provide the possibility of cancelling a task.

1.3.4 Impersonate Grant

The Client Credentials grant is usually used for communication between servers. Communication between Aras Innovator servers requires user information. The token requested by the Client Credentials grant does not contain user information, so the Impersonate grant has been implemented.

Note: Learn more about the Client Credentials grant in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4.4.

The Impersonate grant is a custom grant like the Client Credentials grant. The Impersonate grant is designed to authenticate client by client assertion tokens and provide access tokens by username and database. To request an access token using the Impersonate grant it is necessary to send the client assertion token, scope, database, and username to the token endpoint of the OAuth server.

1. Before users can use the Impersonate Grant with custom applications, users must do the following: Use OpenSSL to generate the certificates pair (PFX, CER files).
2. Put the public certificate (CER file) in the *{Installation_Dir}**OAuthServer**App_Data**Certificates*\ directory.
3. Use a private certificate (PFX file) in the custom application.
4. Configure the new OAuth client based on the InnovatorServer *clientRegistry* node using the allowed impersonate grant located in the *{Installation_Dir}**OAuthServer**OAuth.config* ***directory*.**

Use the configured OAuth client for authentication with the Impersonate grant.

Note: See more information about assertion tokens in JWT specification: https://tools.ietf.org/html/rfc7523

Warning The Impersonate grant is used by the trusted server application.

IClientAssertionProvider is used to provide an assertion token for the ImpersonateTokenProvider. The default implementation of the IClientAssertionProvider is JwtBearerClientAssertionProvider. To create the JwtBearerClientAssertionProvider it is necessary to provide JwtBearerClientAssertionProviderOptions.

```
var certificate = new X509Certificate2(
   privateCertificateFilePath,
   certificatePassword,
   keyStorageFlags);
var assertionProvider = new JwtBearerClientAssertionProvider(
   new JwtBearerClientAssertionProviderOptions
   {
      ClientId = "ClientId",
      Audience = discoveryDocument.Issuer,
      Certificate = certificate,
      AssertionTokenLifetime = TimeSpan.FromSeconds(300)
   });
```

Note: You can load the Certificate in *X509Certificate2* from Certificate Store. For more information see the other contructors of *X509Certificate2* https://docs.microsoft.com/en-us/dotnet/api/system.security.cryptography.x509certificates.x509certificate2.-ctor

JwtBearerClientAssertionProviderOptions have the following options:

- `ClientId` – the OAuth client ID that requests the token.
- `Audience` – determines where the assertion token will be used. In the case of the Impersonate grant, the audience is the OAuth server. To get the correct value use the *Issuer* property of *DiscoveryDocument*.
- `Certificate` – the *X509Certificate2* instance of the client private certificate.
- `AssertionTokenLifeTime` – lifetime of the assertion token. The default value is 300 seconds.

To get the client assertion you must call the *GetClientAssertionAsync* method:

```
ClientAssertion assertion =
```

```
   await assertionProvider.GetClientAssertionAsync();
```

Client assertion contains the following properties:

- `Type` – assertion token type.
- `Value` – assertion token.

To create the ImpersonateTokenProvider it is necessary to pass ImpersonateTokenProviderOptions:

```
// database and username values should be entered by user
String db;
String userName;

var tokenProvider = new ImpersonateTokenProvider(
   new ImpersonateTokenProviderOptions()
   {
      TokenEndpoint = discoveryDocument.TokenEndpoint,
      ClientAssertionProvider = assertionProvider,
      Scope = "Innovator",
      Database = db,
      UserName = userName
   });
```

ImpersonateTokenProviderOptions uses the following options:

- `TokenEndpoint` – URL of OAuth server token endpoint.
- `ClientAssertionProvider` – provider of assertion token.
- `Scope` – scope used to request the token.
- `Database` – name of the database to login to.
- `UserName` – user name.

> **Note:** Custom *IClientAssertionProvider* implementation can be provided. There is no need to provide a password because authentication is performed for client-by-client credentials. In this case client credentials are assertion tokens.

You must call the GetAccessTokenAsync method to get an access token. This method sends a request to the OAuth server token endpoint and returns the access token created by the Impersonate grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

> **Note:** The *GetAccessTokenAsync* method can take the *CancellationToken* as a parameter for cancelling a task.

1.3.5 Authorization Code Grant

The Authorization Code grant type is used to obtain access tokens and is optimized for confidential clients. Since this is a redirection-based flow, the client must be capable of interacting with the resource owner's user-agent (typically a web browser) and capable of receiving incoming requests (via redirection) from the authorization server.

> **Note:** See more about the Authorization Code grant in the OAuth 2.0 RFC specification: https://tools.ietf.org/html/rfc6749#section-4.1

To request an access token using an authorization code grant it is necessary to use the *AuthorizationFlowTokenProvider*. To create the *AuthorizationFlowTokenProvider* it is necessary to create an *INavigator* instance.

Using WpfBrowserNavigator

The default implementation of *INavigator* is *WpfBrowserNavigator* which can be found in Aras.IOM.OAuth.WpfBrowserNavigator nuget package. This navigator is used to allow browser-based login from WPF and WinForms applications. *WpfBrowserNavigator* uses *WebView2* control to process navigation to URLs.

> **Note:** Aras.IOM.OAuth.WpfBrowserNavigator package can be installed from NuGet Package Manager, and also located in the Packages folder within the Aras IOM SDK\<Version>\IOM-<Version>-SDK.zip archive on the FTP. To use Aras.IOM.OAuth.WpfBrowserNavigator.nupkg* from a local source, place it into your NuGet feed (see Appendix A: Local NuGet Repository)

Microsoft Edge WebView2 Runtime is a required component for WebView2 control. WebView2 Runtime must be installed in each machine where developed application is run. It can be downloaded by following link: https://developer.microsoft.com/en-us/microsoft-edge/webview2/#download-section

> **Note:** The Aras.IOM.OAuth.WpfBrowserNavigator package has dependency to Microsoft.Web.WebView2 package. If your project is targeted to net472 and has SDK project type or you use PackageReference to refer to packages in non SDK project type than you need additionally install Microsoft.Web.WebView2 package via Manager of NuGet packages. It is issue of MSBuild and more information is there:
>
> **Note:** https://github.com/dotnet/sdk/issues/20073
>
> **Note:** https://github.com/dotnet/sdk/issues/3031

To create the *WpfBrowserNavigator* it is necessary to call a parameterless constructor. It is possible to define values for displaying a browser window:

```
var navigator = new WpfBrowserNavigator
{
   Width = 900,
   Height = 600,
   Title = "Innovator"
};
```

WpfBrowserNavigator has the following properties:

- `Width` – width of browser window. Default value is 900.
- `Height` – height of browser window. Default value is 600.
- `Title` – title of browser window. Default value is "Innovator".
- `Owner` – owner of browser window in case of using *WpfBrowserNavigator* in WPF application. The browser window will be centered within the owner window. If this property is not set, the browser window will be centered within the current screen working area.
- `OwnerHandle` – owner of browser window in case of using WpfBrowserNavigator in not WPF application. The browser window will be centered within the owner window. If this property is not set, the browser window will be centered within the current screen working area.

Note: The IOM.OAuth.WpfBrowserNavigator.dll is stored next to .NET version of IOM.dll.

To create the AuthorizationFlowTokenProvider it is necessary to pass AuthorizationFlowTokenProviderOptions:

```
// database value should be entered by user
String db;

var tokenProvider = new AuthorizationFlowTokenProvider(
   new AuthorizationFlowTokenProviderOptions()
   {
      DiscoveryDocument = discoveryDocument,
      ClientId = "IOMApp",
      RedirectUrl = "iomapp://token",
      Scope = "Innovator",
      Database = db,
      AuthenticationType = "Windows",
      ResponseMode = ResponseMode.Query,
      Navigator = navigator
   });
```

AuthorizationFlowTokenProviderOptions have the following options:

- `DiscoveryDocument` – instance of *DiscoveryDocument*.
- `GrantType` – a grant type that is used to request an access token. The default value is *AuthorizationCode*. Only the *AuthorizationCode* grant is currently supported.
- `ClientId` – the ID of the OAuth server client that requests a token.
- `RedirectUrl` – URL to return a token.
- `Scope` – scope used to request a token.
- `Database` – name of the database to login to.
- `AuthenticationType` – authentication type to use to authenticate.
- `ResponseMode` – response mode. The Default value is Query.
- `Navigator` – instance of *INavigator*.

Note: `Database` and `AuthenticationType` options are implemented to allow direct login. If both values are set and the required authentication type allows login without the Aras Innovator login page, the OAuth server will try to redirect to the specified authentication type automatically and log the external user into the specified database.

You must call the GetAccessTokenAsync method to get an access token. This method sends a request to the OAuth server Authorize endpoint, receives an authentorization code, sends the authorization code to the token endpoint and gets the access token issued by the Authorization Code grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

Note: The *GetAccessTokenAsync* method can take the *CancellationToken* as a parameter to provide the possibility of cancelling a task.

1.3.6 Authorization Code Grant with Windows Identity Provider

It is possible to use the IOM API for login via Windows authentication without user interaction. To request an access token using Windows authentication you must use the *WindowsTokenProvider*. This provider uses an Authorization Code grant.

To create a *WindowsTokenProvider* it is necessary to provide *WindowsTokenProviderOptions*:

```
// database value should be entered by user
String db;

var tokenProvider = new WindowsTokenProvider(
   new WindowsTokenProviderOptions()
   {
      ClientId = "IOMApp",
      DiscoveryDocument = discoveryDocument,
      RedirectUrl = "iomapp://token",
      Scope = "Innovator",
      AuthenticationType = "Windows",
      Database = db
   });
```

WindowsTokenProviderOptions have the following options:

- `DiscoveryDocument` – an instance of *DiscoveryDocument*.
- `ClientId` – ID of OAuth server client that requests the token.
- `RedirectUrl` – URL to return the token.
- `Scope` – scope used to request the token.
- `Database` – name of the database to login to.
- `AuthenticationType` – the name of the authentication type used for Windows authentication in the OAuth server.
- `MaxRedirectsCount` – maximum number of redirects during authentication. This option is required to limit the number of redirects in a redirection-based flow to prevent eternal redirections. Default value is *10*.

> **Note:** The "iomapp://token" is the default *RedirectUri* value for the IOMApp client of the OAuth server.

You must call the GetAccessTokenAsync method to get an access token. This method sends a request to the OAuth server Authorize endpoint, receives the authorization code, sends the authorization code to the token endpoint, and gets an access token issued by the Authorization Code grant.

```
string accessToken = await tokenProvider.GetAccessTokenAsync();
```

> **Note:** The *GetAccessTokenAsync* method can take *CancellationToken* as a parameter to nake it possible to cancel a task.

2 IOM (COM-Compatible)

This section explains how to use the Aras Innovator Object Model (IOM) in COM-compatible environments, such as VB6, Visual C++, VBA, and Windows Script Host (e.g., VBScript or JScript). It covers basic authentication using username/password, as well as advanced OAuth 2.0 authentication scenarios using token providers and interactive login support.

2.1 Installation and Prerequisites

The Aras Innovator Object Model (IOM) is available in multiple formats to support a variety of development environments and technologies. This section describes how to locate, install, and reference the IOM library for different use cases.

Variant	Target Environment
IOM for COM	COM-compatible environments (VB6, VC++, VBA, VBScript)

2.1.1 IOM for COM

The COM-compatible version is provided to build Windows applications in VB6 or VC++ or write, for example, VBA office macros or Windows Scripting Host applications (VBScript or Jscript) or other environments that rely on COM interop.

A COM-compatible version of the IOM.dll can be found in the `COM` folder within the `Aras IOM SDK\<Version>\IOM-<Version>-SDK.zip` archive on the FTP.

To register and use:

To use the DLL, it must be registered in the Windows Registry. Use the following procedure:

5. Copy IOM.dll into a local folder.
6. Register the DLL using the appropriate command:

For x86: `%windir%\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe .\IOM.dll /codebase`

For x64: `%windir%\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe .\IOM.dll /codebase`

Note: IOM type library (IOM.tlb) can be found near IOM.dll. It is recommended to use type library provided with IOM.dll instead of generating new one with /tlb argument for RegAsm.exe tool.

7. If the assembly was successfully registered it must appear in the list of available COM references as "Aras IOM 14.0 API (COM-compatible)".
8. Add the reference to yours project:

```
#import "{SDK_Dir}\COM\IOM.tlb"
```

2.2 Basic Connection using Username/Password

Before interacting with Aras Innovator via the COM-compatible IOM, your application must establish a connection to the Aras server and perform a login. If authentication succeeds, you can create an Innovator object instance using the IOM factory. This object is the main entry point for submitting AML-based requests and processing responses.

This section demonstrates how to connect using a standard server URL, database name, username, and password in a COM-based environment such as VC++.

Basic Connection Flow

1. Create an HttpServerConnection using the server URL, database name, username, and password.
2. Call the Login() method on the connection.
3. If login is successful, use IomFactory.CreateInnovator() to create an Innovator instance.
4. Use the Innovator instance to interact with Aras Innovator via IOM.
5. When done, call Logout() on the connection to cleanly end the session.

Sample Code

```
_bstr_t url = L"https://myserver/MyInnovator/Server/InnovatorServer.aspx";
// database, username and password values should be entered by user
_bstr_t db;
_bstr_t userName;
_bstr_t password;

IOM::IIomFactoryComIncomingPtr iomFactory(__uuidof(IOM::IomFactory));
IOM::IHttpServerConnectionComIncomingPtr connection =
   iomFactory->CreateHttpServerConnection(url, db, userName, password);

IOM::IItemComIncomingPtr loginResult = connection->Login();

if (loginResult->isError())
{
   std::cout << "Failed to login.";
}
else
{
   IOM::IInnovatorComIncomingPtr innovator =
      iomFactory->CreateInnovator(connection);
}
```

Logging Out

It is strongly recommended to explicitly log out of Aras Innovator when your application is finished.

```
…
conn->Logout();
…
```

2.3 Obtaining an Access Token from COM Applications

2.3.1 OAuthFactory for creating COM objects

The COM-compatible IOM API introduces *OAuthFactory* class. Using this factory class users can create a COM instance of discovery document providers or token providers.

OAuthFactory is a CoClass with "Aras.IOM.OAuthFactory.12.0" progId, that implemets *IOAuthFactory* interface with the following methods:

```
IDiscoveryDocumentProvider* CreateDiscoveryDocumentProvider(BSTR
innovatorServerUrl)
```

The method creates a *DiscoveryDocumentProvider* COM object based on the Innovator server URL.

```
IPasswordTokenProviderOptions* CreatePasswordTokenProviderOptions()
```

The method creates a *PasswordTokenProviderOptions* COM object which will be used while creating the *PasswordTokenProvider*.

```
ITokenProvider* CreatePasswordTokenProvider(IPasswordTokenProviderOptions*
options)
```

The method creates a *PasswordTokenProvider* COM object based on the provided *IPasswordTokenProviderOptions*.

```
IWindowsTokenProviderOptions* CreateWindowsTokenProviderOptions()
```

The method creates a *WindowsTokenProviderOptions* COM object which will be used while creating the *WindowsTokenProvider*.

```
ITokenProvider* CreateWindowsTokenProvider(IWindowsTokenProviderOptions*
options)
```

The method creates a *WindowsTokenProvider* COM object based on the provided *IWindowsTokenProviderOptions*.

```
IAuthorizationFlowTokenProviderOptions*
CreateAuthorizationFlowTokenProviderOptions()
```

The method creates an AuthorizationFlowTokenProviderOptions COM object which will be used for AuthorizationFlowTokenProvider creation.

```
ITokenProvider*
CreateAuthorizationFlowTokenProvider(IAuthorizationFlowTokenProviderOption
s* options)
```

The method creates an AuthorizationFlowTokenProvider COM object based on the provided IAuthorizationFlowTokenProviderOptions.

Note: Option objects are used to pass arguments to factory methods creating corresponding provider objects. Developer should set property values of option object as necessary before passing it to corresponding factory method.

Here is an example of creating *PasswordTokenProvider* with *OAuthFactory* in C++:

```
_bstr_t innovatorServerUrl =
   L"https://myserver/MyInnovator/Server/InnovatorServer.aspx";
// database, username and password values should be entered by user
_bstr_t db;
_bstr_t userName;
_bstr_t password;
```

```
IOM::IOAuthFactoryPtr oauthFactory(__uuidof(IOM::OAuthFactory));

IOM::IDiscoveryDocumentProviderPtr discoveryDocumentProvider =
   oauthFactory->CreateDiscoveryDocumentProvider(innovatorServerUrl);
IOM::IDiscoveryDocumentPtr discoveryDocument =
   discoveryDocumentProvider->GetDiscoveryDocument();

IOM::IPasswordTokenProviderOptionsPtr options =
   oauthFactory->CreatePasswordTokenProviderOptions();
      options->TokenEndpoint = discoveryDocument->GetTokenEndpoint();
      options->ClientId = L"IOMApp";
      options->Scope = L"Innovator";
      options->Database = db;
      options->UserName = userName;
      options->Password = password;

IOM::ITokenProviderPtr passwordTokenProvider =
   oauthFactory->CreatePasswordTokenProvider(options);
```

2.3.2 COM-Compatible WpfBrowserNavigator

A COM-compatible version of the IOM.OAuth.WpfBrowserNavigator.dll can be found near the COM-compatible IOM.dll in the `COM` folder within the `Aras IOM SDK\<Version>\IOM-<Version>-SDK.zip` archive on the FTP.

This assembly contains classes that provide the possibility of using browser-based authentication for applications in VB6, VC++, VBScript, JScript etc.

To use the WpfBrowserNavigator DLL it must be registered in the same way as the IOM.dll. See steps for registration in section 10.2.

> **Note:** WpfBrowserNavigator type library (IOM.OAuth.WpfBrowserNavigator.tlb) can be found in the COM folder within the Aras IOM SDK\Aras-IOM-<Version>-SDK.zip archive and it is recommended to use it in the same way as IOM.COM type library.

Here is the sample code for creating *WpfBrowserNavigator* in C++:

```
// windowHandle should be set by developer
long windowHandle;

IOM_OAuth_WpfBrowserNavigator::IWpfBrowserNavigatorPtr wpfBrowserNavigator
   (__uuidof(IOM_OAuth_WpfBrowserNavigator::WpfBrowserNavigator));

wpfBrowserNavigator->Height = 600;
wpfBrowserNavigator->Width = 980;
wpfBrowserNavigator->Title = L"Innovator";
wpfBrowserNavigator->OwnerHandle = windowHandle;
```

> **Note:** The browser window will be centered within the owner window specified by the OwnerHandle property. If this property is not set, the browser window will be centered within the current screen working area.

2.3.3 Using Token Provider with HttpServerConnection

The *IomFactory* class has a public *CreateHttpServerConnection2* method for creating a connection to the Aras Innovator server based on the provided *ITokenProvider* implementation.

CreateHttpServerConnection2 method takes the following parameters:

- `innovatorServerUrl` – URL to Innovator server.
- `tokenProvider` – instance of ITokenProvider.
- `protocolType` – information about protocol that is used by the OAuth server.

Here is an example of creating an HttpServerConnection with AuthorizationFlowTokenProvider in C++:

```
IOM::IOAuthFactoryPtr oauthFactory(__uuidof(IOM::OAuthFactory));

_bstr_t innovatorServerUrl =
   L"https://myserver/MyInnovator/Server/InnovatorServer.aspx"

IOM::IDiscoveryDocumentProviderPtr discoveryDocumentProvider =
   oauthFactory->CreateDiscoveryDocumentProvider(innovatorServerUrl);
IOM::IDiscoveryDocumentPtr discoveryDocument =
   discoveryDocumentProvider->GetDiscoveryDocument();

// windowHandle should be set by developer
long windowHandle;

IOM_OAuth_WpfBrowserNavigator::IWpfBrowserNavigatorPtr wpfBrowserNavigator
   (__uuidof(IOM_OAuth_WpfBrowserNavigator::WpfBrowserNavigator));

wpfBrowserNavigator->Height = 600;
wpfBrowserNavigator->Width = 980;
wpfBrowserNavigator->Title = L"Innovator";
wpfBrowserNavigator->OwnerHandle = windowHandle;

IOM::IAuthorizationFlowTokenProviderOptionsPtr options =
   oauthFactory->CreateAuthorizationFlowTokenProviderOptions();
options->DiscoveryDocument = discoveryDocument;
options->ClientId = L"IOMApp";
options->Scope = L"Innovator";
options->RedirectUri = L"iomapp://token/";
options->ResponseMode = IOM::ResponseMode::ResponseMode_Query;
options->Navigator = wpfBrowserNavigator;

IOM::ITokenProviderPtr tokenProvider =
   oauthFactory->CreateAuthorizationFlowTokenProvider(options);

IOM::IHttpServerConnectionComIncomingPtr connection =
   iomFactory->CreateHttpServerConnection2(
      innovatorServerUrl,
      tokenProvider,
      discoveryDocument->GetProtocolType());
```

3 Working with the IOM API

Once a connection to Aras Innovator has been established using the IOM library, the next step is to interact with the system through the IOM API, primarily by working with Item objects and using AML (Aras Markup Language).

This section provides an overview of the IOM Object Model and describes how to create, query, and modify data using the core classes and methods provided by the library.

> **Note:** Code examples in this section are written in C#. While direct COM examples are not included, the concepts and object model apply similarly in COM environments (e.g., VB6, VC++, VBA). Developers using COM should interpret these examples as logical equivalents and refer to the COM API interfaces accordingly.

3.1 Overview of the IOM Object Model

This section provides a general description of the IOM (Innovator Object Model or Item Object Model) API. A more detailed API reference may be obtained from one of the following:

- **On-line**: Go to https://www.aras.com/support/documentation/
  Under the section **Other Documents**, click **On-Line API Guide.html**.
- **From Aras Innovator Client UI**: Login as Innovator Administrator. Under the Help menu, select API Reference.
- **In Aras Innovator CD Image**: Go to the documentation folder and open `Aras Innovator - API Guide.html`.
- **Reference to Aras Innovator Programmers Guide** – it will have description of AML

The Aras Innovator Object Model (IOM) is a lightweight, flexible API designed to abstract and simplify communication with the Aras Innovator server. It allows you to build, send, and parse AML documents using object-oriented constructs such as Item, Connection, and Innovator.

Object Model Concepts

Concept	Description
Item	The central object in IOM, representing a unit of data (corresponding to an ItemType). Can represent a single item, a collection of items, or an error.
Innovator	The main entry point for creating and manipulating Item objects. Provides factory methods and helper functions for interacting with the Aras database.
Connection	Represents an active connection to the Aras Innovator server, used to authenticate and send AML requests.

The IOM is an Object Model for the AML, but it is not purely Object Oriented. Using Object Oriented terms, an ItemType is like a 'Class' and the Item is like an 'Object'. Although the Item is an Object with methods, there is only one Item Class for all ItemTypes. In a pure Object-Oriented representation, there

would be a Class for each ItemType because each ItemType has its own set of Properties to describe the different Items.

An Item Object is intended to be abstract and pliable. Depending on its internal structure, the Item Object usually represents one of five following supported types of IOM Items:

- **Single** Aras Innovator Item of an arbitrary ItemType.
- **Set** of Aras Innovator Items, as in the case for results of the action 'get' when more than one Item returns.
- **Error**, as in the case when an action request results in an error from the Aras Innovator Server.
- **Result** to represent an arbitrary text wrapped by <Result> XML tags.
- **Logical** to represent a set of properties wrapped in a logical statement by one of logical XML tags: <or>, <and>, or <not>; usually used to specify the search criteria for the action 'get'.

The IOM is intended to be a generic and compact API for modeling the Item structure of the AML as abstract Objects. Most of the methods for the IOM deal with memory management of the AML document for the Item Object. The AML is a script sent as a message to the Aras Innovator Server. The IOM is an Object API to build the AML messages, submit them to the Aras Innovator Server and parse an AML document that is returned by the Server.

While the IOM API internally constructs and parses AML XML, you do not need to write raw AML for common tasks. Instead, the IOM exposes object methods that generate valid AML under the hood.

For example, creating a Part looks like this in C#:

```
Item part = innovator.newItem("Part", "add");
part.setProperty("item_number", "C123");
part.setProperty("name", "Capacitor");
part.setProperty("classification", "Electronics");
Item result = part.apply();
```

Which is equivalent to this AML:

```
<Item type="Part" action="add">
  <item_number>C123</item_number>
  <name>Capacitor</name>
  <classification>Electronics</classification>
</Item>
```

There are two major types of methods in the Item Class:

- methods that only work with an item's AML in memory.
- methods that communicate with the Server, i.e., send request(s) to and get response(s) from the server.

All get\set type of methods as well as isXXX(...) (e.g. isError(), isCollection(), etc.) and add\remove methods (e.g. addRelationship(...), removeProperty(...), etc.) belong to the former group. Methods like fetchXXX(...) (e.g. fetchLockStatus()), apply(...), email(...), promote(...), lock\unlockItem(...), etc. belong to the latter group. In case a method sends a request to the server it must be explicitly mentioned in the API reference method comments.

3.2 Innovator Class

In Innovator Class, methods like `applyAML(…)`, `applyMethod(…)`, and `applySQL(…)`, being type Item, send the apply request to the Aras Innovator Server. In response, the Server creates XML that apply_ methods use to construct an Item Object to return; methods like `getItemById(…)` and `getItemByKeyedName(…)` search the database the logged in session is connected to, and methods like `newItem(…)`, `newError(…)` and `newResult(…)` construct a new instance of the Item Object.

Other members of this class perform miscellaneous non-Item related operations. Use them if you need to get a new GUID (methods `getNewID()`), generate a next sequence value (method `getNextSequence(…)`), or calculate the MD5 hash value for a given string (`method ScalcMD5(…)` .)

3.3 Item Class

An Item Object can represent an Item, a set of Items, or an Error.

The Item public constructor has one required argument itemtype-name and one optional argument action. The new Item is only populated with the Properties and default values from the ItemType when the optional action argument is 'add'.

The Item Class public field `dom` represents a DOM Object that holds the data for the Item in the AML format.

3.3.1 Base Methods

The Item Class base method `apply(…)` submits the AML apply request to the Aras Innovator Server using the context Item DOM as the AML source and returns a new Item built on the XML returned by the Server. In contrast, the base method `loadAML(…)` does not return a new Item but rather rebuilds `this.dom` using AML taken as an argument.

You can call the `clone(…)` method to get a new identical instance of the context Item and you can call the `setNewID()` method to replace the context Item id by the newly generated GUID.

3.3.2 Boolean Methods

Use Call method `isCollection()` to find out whether or not the Item represents a set of Items, e. g. whether or not its dom property holds more than one Item node. Use Call `isError()` method to find out whether or not the Item represents an Error. See the IOM API on-line reference for more boolean members of the Item Class.

3.3.3 Attribute Methods

Methods such as `getAction()`, `getID()`, and `getType()` return a value of Item node action, id, and type attribute respectively, while method `getAttribute(…)` takes an attribute-name as an argument and, therefore, can be used to get any attribute by name. Thus, `getAction()` is a short cut to `getAttribute("action")`, `getID()` is a short cut to `getAttribute("id")`, and so on.

Each get method in this group has a corresponding set method: `setAction(…)`, `setID(…)`, `setType(…)`, and `setAttribute(…)`. Notice that method `setAttribute(…)` not only sets a new value for an existing attribute but can also add a new attribute to the Item node and then sets its value.

3.3.4 Property Methods

Property methods comprise a set of accessors to Item properties and Item property's attributes: get_, set_ (which acts also as add_), and remove_. In addition, if a property has/needs a nested Item there are methods to get/insert these nested Items.

Accessors to Properties are `getProperty(…)`, `setProperty(…)`, and `removeProperty(…)`.These methods take the property name as an argument.

Accessors to Property's attribute are `getPropertyAttribute(…)`, `setPropertyAttribute(…)`, and `removePropertyAttribute(…)`. These methods take property and attribute names as arguments.

The method `setProperty(…)/setPropertyAttribute(…)` not only sets a new value for an existing Item property/ property's attributes but creates new property/ property's attributes and then sets its value if the property/ property's attributes with a given name does not yet exist.

The `setProperty` method requires property values to be in a locale-neutral format. Decimal and float values should use the period symbol (.) as the decimal separator and no digit separator (i.e. commas separating thousands). The dash symbol (-) should be used to denote a negative value. Date/Time values should be in 'YYYY-MM-DD[Thh:mm:ss]' format and in the local (or corporate) time zone. Language-specific values should be set using the language code as the third argument.

Accessors to a Property's nested Item are: `getPropertyItem(…)`, `setPropertyItem(…)`, and `createPropertyItem(…)`. All these methods take property name as an argument, and method `setPropertyItem(…)` needs, in addition, an Item Object as the second argument to create a DOM for the nested Item.

3.3.5 Relationship Methods

Relationship methods are made up of a set of Item's relationship accessors: `getRelationships(…)`, `addRelationship(…)`, `createRelationship(…)`, and `removeRelationship(…)`. The difference between `createRelationship(…)` and `addRelationship(…)`, two methods that both are adding an Item node to the Relationship parent node, is subtle. Let's consider the following server-side C# method to illustrate how `createRelationship(…)` works:

```
Item myItem = innovator.newItem("myType","myAction");
myItem.createRelationship("User","add");
```

This code results in the following myItem.dom XML:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction">
  <Relationships>
    <Item isNew="1" isTemp="1" type="User" action="add" />
  </Relationships>
</Item>
```

Similar code that exercises addRelationship(…) methods is:

```
Item myItem = innovator.newItem("myType","myAction");
Item relItem = innovator.newItem("User","add");
myItem.addRelationship(relItem)
```

And this code results in the following myItem XML:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction">
  <Relationships>
    <Item isNew="1" isTemp="1" type="User" action="add"
id="7EA3F18935CC493A900DAB63E839FDA2">
      <classification>/*</classification>
      <default_vault>67BBB9204FE84A8981ED8313049BA06C</default_vault>
    </Item>
  </Relationships>
</Item>
```

As you can see `addRelationship(…)` produces a more detailed Item node under the Relationship parent node.

There are three methods `getRelatedItem(…)`, `setRelatedItem(…)`, and `createRelatedItem(…)` that are valid for relationship Items only and are used as a short cut to the `Item.get/setPropertyItem()` methods.

3.3.6 Item Collection Methods

This group is comprised of methods to work with collections. Collection in this context is a set of Items as in the case for results from a query. The method `appendItem(…)` appends its Item Object argument to an existing collection or converts a single Item instance to a set of Items. This mechanism is illustrated in the following C# sample:

```
Item myItem  = innovator.newItem("myType","myAction");
```

At this point, myItem presents a single Item instance, myItem.dom XML looks like this:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction" />
and myItem.isCollection() returns false.  But three extra line of code:
Item addedItem  = innovator.newItem("added","myAction");
// set ID to be able to remove addedItem later
addedItem.setID(innovator.getNewID());
myItem.appendItem(addedItem);
```

It converts myItem to a collection, e.g. `myItem.isCollection()` returns `true`, and `myItem.dom` XML becomes as follows:

```
    <AML>
      <Item isNew="1" isTemp="1" type="myType" action="myAction" />
      <Item isNew="1" isTemp="1" type="added" action="myAction"
id="B12F9384B1DE4C4F8158C36D18269BE9" />
    </AML>
```

If the Item object id is not NULL it can be removed from the collection:

```
myItem.removeItem(addedItem);
```

Use the `getItemCount()` method to determine the size of a collection, the `getItemByIndex(…)` method to get an instance of the Item Object based on its position inside of the collection, and method `getItemsByXPath(…)` to find an Item by its XPath.

3.3.7 Logical Methods

The methods `newOR()`, `newAND()`, and `newNOT()` insert logical nodes with tag <or>, <and> and <not> respectively under the parent Item node and returns an Item Object that represents a newly inserted logical node. For example, the following code:

```
Item myItem  = innovator.newItem("myType","myAction");
Item logicalOR = myItem.newOR();
logicalOR.setProperty("foo", "bar");
```

produces the following `myItem.dom` XML:

```
<Item isNew="1" isTemp="1" type="myType" action="myAction">
  <or>
    <foo>bar</foo>
  </or>
</Item>
```

The method `removeLogical(…)` removes a logical node specified by the method's argument.

3.3.8 Creating New Item Method

You can create a new item using method `newItem(…)`. See code samples in sections 3.3.6-3.3.87 for the method usage illustration.

3.3.9 Error Methods

Error methods are comprised of a set of accessors to Error specific properties such as "faultcode" (methods `get/setErrorCode(…)`,) "faultstring" (methods `get/setErrorString(…)` ,) "faultactor" (`get/setErrorSource(…)` ,) and "detail" (methods `get/setErrorDetail(…)` .)

The code sample below:

```
Item error = innovator.newError("default detail");
error.setErrorCode("any number");
error.setErrorString("Hello, World");
error.setErrorSource("myMethod");
error.setErrorDetail("new detail");
```

produces the following Error Item DOM:

```
<Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Body>
    <Fault>
      <faultcode>any number</faultcode>
      <faultstring>Hello, World</faultstring>
      <faultactor>myMethod</faultactor>
      <detail>new detail</detail>
    </Fault>
  </Body>
</Envelope>
```

3.3.10 Extended Item Class Methods

This set of methods implements specific functionality on the Item, which extends the base Item Class. For reference purposes all the Extended Item Class methods are organized in the following four categories:

- Getting Innovator reference method:
  `getInnovator()` – see examples of usage in section 3.2.
- Lock methods:
  `lockItem()`
  `unlock`Item()
- Life Cycle method:
  `promote(…)`
- Workflow methods:
  `instantiateWorkflow(…)`

The following methods are obsolete and will be removed from future releases:

`startWorkflow(…)` –use `Item.apply("startWorkflow")` instead.

`cancelWorkflow(…)` - use `Item.apply("CancelWorkflow")` instead.

`closeWorkflow(…)` - use `Item.apply("closeWorkflow")` instead.

The detailed characteristic and usage illustration of these methods is outside the scope of the document. See the on-line API reference for more details.

4 IOM Performance Best Practices

Efficient use of the Aras IOM library is essential for responsive applications, especially when working with large datasets, network latency, or high-volume systems. This section outlines best practices to optimize performance when using IOM in .NET or COM applications.

4.1 Use select Attribute to Limit Returned Data

By default, get actions in AML return all properties of the item, which often leads to unnecessarily large XML responses and increased server-side processing.

To avoid this, always specify which properties you need using the select attribute:

```
Item query = innovator.newItem("Part", "get");
query.setAttribute("select", "item_number,name");
// set properties criteria for filtering if appropriate
Item result = query.apply();
```

You can also request specific fields from related items:

```
Item query = innovator.newItem("Part", "get");
query.setAttribute("select", "id,team_id(id,name)");
// set properties criteria for filtering if appropriate
Item result = query.apply();
```

4.2 Use Pagination for Large Queries

When querying collections that may contain many items (e.g., grid views, dashboards), use page and pagesize attributes to retrieve only a subset of results:

```
Item query = innovator.newItem("Part", "get");
query.setAttribute("page", "1");
query.setAttribute("pagesize", "100");
query.setAttribute("returnMode", "itemsOnly");
// set properties criteria for filtering if appropriate
Item result = query.apply();
```

Pagination keeps memory usage low and improves client responsiveness.

4.3 Use returnMode in Paginated AML Queries

The returnMode attribute allows you to control how much data is returned in response to an AML query. When working with paginated queries, setting returnMode appropriately can significantly improve performance by reducing database load and the size of the response.

The three possible values for returnMode in paginated AML queries are as follows:

- countAndItems: (Default) Returns both the paginated set of items and the total count.
- itemsOnly: Returns only the items; skips the total count query to the database.
- countOnly: Returns only the total count; does not return any items.

In case you use returnMode='itemsOnly' with pagination – server will not send count request to DB to retrieve max count of items for current request:

```
Item query = innovator.newItem("Part", "get");
query.setAttribute("select", "item_number,name");
query.setAttribute("page", "1");
query.setAttribute("pagesize", "100");
query.setAttribute("returnMode", "itemsOnly");

Item result = query.apply();
```

In case you use returnMode='countOnly' – server will send only count request to DB:

```
Item query = innovator.newItem("Part", "get");
// set properties criteria
query.setAttribute("returnMode", "countOnly");
Item result = query.apply();
XmlNode itemMax =
part.dom.SelectSingleNode(".//Message/event[@name='itemmax']/@value");
```

4.4 Minimize the Number of Server Requests

The IOM library provides many helper methods for accessing and manipulating individual items. However, when working with collections of items, using these methods naively can result in one server request per item, leading to unnecessary overhead and degraded performance.

Reducing the number of HTTP requests is one of the most important performance optimizations you can make when working with the Aras IOM. Each call to the server introduces latency, server-side processing overhead, and possible authentication and validation steps.

The key to efficient IOM usage is to favor batch-oriented operations that operate on multiple items in a single AML request.

Be Aware of Server-Triggering Methods

Certain IOM methods, especially those on Item and Innovator, perform server calls under the hood. When called inside a loop or per record, this can lead to dozens or hundreds of server round-trips.

Examples of Methods That Trigger Server Requests:

API Object	Method
Innovator	applyAml, applySql, getItemById, getItemByKeyedName, getAssignedActivities, getNextSequence, getUserAliases, getFileUrl, getFileUrls, ConsumeLicense
Item	apply, fetchDefaultPropertyValues, fetchLockStatus, fetchRelationships, lockItem, unlockItem

Common Pitfall: One Call per Item

✗ Inefficient Pattern:

```
for (int i = 0; i < items.getItemCount(); i++)
{
    Item itm = items.getItemByIndex(i);
    Item lockStatus = itm.fetchLockStatus(); // One request per item
}
```

This results in N requests to the server if the collection contains N items.

Preferred Pattern: Batched AML

✅Efficient Pattern: Build a targeted AML query to retrieve everything you need in one request:

```
Item query = innovator.newItem("Part", "get");
query.setAttribute("select", "id,locked_by_id,state");
query.setProperty("item_number", "123%", "like"); // example filter
Item result = query.apply();
```

One server request retrieves all necessary fields

4.5 Reducing Response Size with doGetItem="0"

When performing actions like add, update, edit, or lock where you do not need to handle the response Item, you can improve performance by including the doGetItem="0" attribute. This instructs the server not to return a full result, reducing processing time and response size.

C#

```
Item item = innovator.newItem("Part", "edit");
item.setID("1234567890ABCDEF");
item.setProperty("name", "Updated Part Name");
item.setAttribute("doGetItem", "0");

Item result = item.apply();
if (result.isError())
{
    // Handle error
}
```

5 Cookbook

This section is a Cookbook of recipes to help you solve common tasks while developing Methods. The examples are shown in C# when possible.

5.1 Create an Aras Innovator Object

Many operations in IOM require an instance of the Innovator class, which serves as the primary entry point for constructing AML queries, executing actions, and working with Item objects.

Technique

To create an Innovator object, you need a valid IServerConnection instance (e.g., HttpServerConnection).Use the factory method provided by IOM.

C#

```
Innovator myInnovator = IomFactory.CreateInnovator(connection);
```

5.2 Create an Item Object

You need an Item Object to submit a query or to add an Item.

Technique

There is a way to create a new Item Object; by calling the factory methods on *Innovator* object.

C#

```
Item myItem   = myInnovator.newItem(myType,myAction);
Item myResult = myInnovator.newResult(resultText);
Item myError  = myInnovator.newError(errorMessage);
```

5.3 Query for an Item

You want to query for an Item that you know by id and type.

Technique

There are a few ways to get an Item when you know its id and type, the simplest being the `Innovator.getItemById()` method. However, if you need to be granular about your request then building the query using the IOM is required. This provides the ability to include controls to limit the results and define the structure to be returned for the Items found.

C#

```
Item qryItem = myInnovator.newItem(myType,"get");
qryItem.setID(myId);
Item results = qryItem.apply();

Item results = myInnovator.getItemById(myType, myId);
```

5.4 Check for Errors After Server-Triggered Calls

You want to ensure that server-triggering methods returned a successful result before using the returned Item.

Technique

Any method that causes a server round trip — such as apply(), getItemById(), or fetchRelationships() — may return an error Item. Failing to check for errors can lead to null references, incorrect logic flow, or silent failures.

The best practice is to always call .isError() on the returned Item and handle it accordingly.

```
Item part = myInnovator.getItemById("Part", "1234");
if (part.isError())
{
    throw new Exception("Failed to retrieve Part: " + part.getErrorDetail());
}
```

An Item with isEmpty() == true means the server returned no matching items (getErrorCode() == "0"), which is not necessarily an error — just a signal that nothing was found:

```
Item partQuery = myInnovator.newItem("Part", "get");
partQuery.setProperty("item_number", "1234");

Item result = partQuery.apply();

if (result.isError() && !result.isEmpty())
{
    throw new Exception("Query failed: " + result.getErrorDetail());
}

if (result.isEmpty())
{
    // Nothing matched the query
}
```

5.5 Query and Iterate Over a Set of Items

You want to query for the Items that match some criteria and generate an HTML Table as the result.

Technique

There is no difference in setting up a query for a single Item or for many. Only the criteria define the set size returned. In this recipe you create an Item and populate the query criteria, apply it, and iterate over the Items returned producing an HTML `<TABLE>` fragment.

C#

```
Item qryItem = myInnovator.newItem("Part","get");
qryItem.setAttribute("select","item_number,description,cost");
qryItem.setProperty("cost", "100");
qryItem.setPropertyCondition("cost", "gt");
Item results = qryItem.apply();
int count = results.getItemCount();
int i;
string content = "<table>";
for (i=0; i<count; ++i)
{
  Item item = results.getItemByIndex(i);
  content += "" +
    "<tr>" +
      "<td>" + item.getProperty("item_number") + "</td>" +
      "<td>" + item.getProperty("description") + "</td>" +
      "<td>" + item.getProperty("cost") + "</td>" +
    "</tr>";
}
content += "</table>";
Item result = myInnovator.newResult(content);
```

5.6 Query for an Item and Return Its Configuration

You want to query for an Item and return its configuration in the results.

Technique

To query for an Item and retrieve its structure you build the query as the structure you want returned. Use the IOM methods to add the relationships you want and build the structure in the Item. The server returns the structure that follows the request structure.

This recipe illustrates several related concepts together, which are how to get a set of Items from an Item and how to iterate over the set, plus how to get the related Item from the relationship Item.

C#

```
// Set up the query Item.
Item qryItem = myInnovator.newItem("Part","get");
qryItem.setAttribute("select","item_number,description,cost");
qryItem.setID(myId);

// Add the BOM structure.
Item bomItem = myInnovator.newItem("Part BOM","get");
bomItem.setAttribute("select","quantity,
related_id(item_number,description,cost)");
```

```
qryItem.addRelationship(bomItem);

// Perform the query.
Item results = qryItem.apply();

// Test for an error.
if (results.isError()) {
  return innovator.newError("Item not found: " + results.getErrorDetail());
}

// Get a handle to the BOM Items.
Item bomItems = results.getRelationships();
int count = bomItems.getItemCount();
int i;

// Create the results content.
string content = "<table border='1'>" +
  "<tr>" +
    "<td>Part Number</td>" +
    "<td>Description</td>" +
    "<td>Cost</td>" +
    "<td>Quantity</td>" +
  "</tr>";

// Iterate over the BOM Items.
for (i=0; i<count; ++i)
{
// Get a handle to the relationship Item by index.
  Item bom = bomItems.getItemByIndex(i);
// Get a handle to the related Item for this relationship Item.
  Item bomPart = bom.getRelatedItem();

  content += "" +
    "<tr>" +
      "<td>" + bomPart.getProperty("item_number") + "</td>" +
      "<td>" + bomPart.getProperty("description") + "</td>" +
      "<td>" + bomPart.getProperty("cost") + "</td>" +
      "<td>" + bom.getProperty("quantity") + "</td>" +
    "</tr>";
}
content += "</table>";

Item result = myInnovator.newResult(content);
```

5.7 Query Using AML to Construct the Query Criteria

You want to perform a query using the AML to construct the query criteria.

Technique

Create an Item Object but use the `Item.loadAML()` method to populate the Item.

C#

```
var qryItem = myInnovator.newItem();
qryItem.loadAML(
```

```
  "<Item type='Part' action='get' select='item_number,description,cost'>" +
    "<item_number condition='like'>1%</item_number>" +
    "<Relationships>" +
      "<Item type='Part BOM' action='get' select='quantity'>" +
        "<quantity condition='gt'>1</quantity>" +
      "</Item>" +
    "</Relationships>" +
  "</Item>"
);

var resultItem = qryItem.apply();
if (resultItem.isError()) {
  // handle error here
  return;
}

var count = resultItem.getItemCount();
for (i=0; i<count; ++i) {
  var item = resultItem.getItemByIndex(i);
}
```

5.8 Query Using Relationships as Search Criteria

You want to search for an Item using the relationship and related Items as search criteria. In this recipe, we get the User Item that matches the Identity name as an Alias relationship to the User Item.

Technique

The following are some key points to understand when constructing an AML query:

1. Use the get action on the relationship Items to include it as search criteria.
   a. Without the get action the relationship Item is ignored as search criteria.
   b. The relationship Items are also returned. Currently there is no way to use relationships as search criteria and not return them in the results, as you can with the related Item described below.
   2. Include the related_id property name in the select attribute for the relationship Item if you want to return the related Item nested inside the related_id property in the results.

      `<Item type="Part BOM" select="quantity,related_id"/>`

      Use () to include the select attribute value for the related Item inside the select attribute for the relationship Item.

      `<Item type="Part BOM" select="quantity,related_id(item_number,description)"/>`
   3. The select attribute for the nested Item tag for the related_id property has higher precedence over the select value inside the () for the relationship's select attribute.
   4. The get action is not required for the nested Item tag for the related_id property to include it as search criteria.

      These two AML scripts are equivalent queries for selecting the name property for the related Item:

```
<Item type="User" action="get" select="first_name,last_name,email">
  <Relationships>
    <Item type="alias" action="get" select="related_id(name)"/>
  </Relationships>
</Item>
```

```
<Item type="User" action="get" select="first_name,last_name,email">
  <Relationships>
    <Item type="alias" action="get" select="related_id">
      <related_id>
        <Item type="Identity" action="get" select="name"/>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

      Clearly the first example is simpler and requires less coding (referring to the IOM logic that would construct the AML) and is the recommended style when all you require is specifying a select for the related Item for the query.
      But the second style opens the opportunity to now include additional search criteria for the related Item.

AML

```
<Item type="User" action="get" select="first_name,last_name,email">
  <Relationships>
    <Item type="Alias" action="get" select="related_id">
<!--
This get will limit root Items to only those that match the relationship
criteria.
The get action is required otherwise the criteria are ignored.
```

```
To include the nested Item tag for the related_id include the property name
in the select attribute for the relationship Item.
Can include the select attribute value for the related Item inside ()
i.e. related_id(name)
-->
      <related_id>
        <Item type="Identity" action="get" select="keyed_name">
<!--
This get has no effect and the search will work with or without it.
It is recommended that you include it because the AML parser may be stricter
in the future.
The select attribute over rules the parent relationships select.
-->
          <name>Larry Bird</name>
        </Item>
      </related_id>
    </Item>
  </Relationships>
</Item>
```

C#

```
var qry = myInnovator.newItem("User","get");
qry.setAttribute("select","first_name,last_name,email");

var alias = myInnovator.Item("Alias","get");
alias.setAttribute("select","related_id");

var identity = myInnovator.Item("Identity","get");
identity.setAttribute("select","name");
identity.setProperty("name", "Larry Bird");

alias.setRelatedItem(identity);
qry.addRelationship(alias) ;

var results = qry.apply();
if (results.isError()) {
  // handle error here
  return;
}
```

5.9 Recursive Query on a Related Item

You want to perform a recursive query on a related item.

Technique

Use the GetItemRepeatConfig action to perform the query. For more information, refer to the Aras Innovator Programmers Guide.

AML

```
<Item type="Part" action="GetItemRepeatConfig"
select="item_number,name,major_rev,has_change_pending,state" id="{@id}">
<Relationships>
   <Item type="Part BOM" select="sort_order,quantity,_kit_flag,related_id"
repeatProp="related_id" repeatTimes="6"/>
</Relationships>
</Item>
```

5.10 Add an Item Configuration in One Transaction

You want to add an Item configuration like a BOM as one transaction.

Technique

Adding an Item configuration is done by building the Item structure using the IOM methods.

C#

```
var partItem = myInnovator.newItem("Part","add");
partItem.setProperty("item_number", "123-456");
partItem.setProperty("description", "Blah blah");

var bomItem = myInnovator.Item("Part BOM","add");
bomItem.setProperty("quantity", "10");

var relatedItem = myInnovator.Item("Part","get");
relatedItem.setProperty("item_number", "555-555");

bomItem.setRelatedItem(relatedItem);
partItem.addRelationship(bomItem) ;

var resultItem = partItem.apply();
if (resultItem.isError()) {
  // handle error here
  return;
}
```

Technique

The following is the same thing but uses the *Item*.loadAML() method to populate the Item Object with AML text.

C#

```
var partItem = myInnovator.newItem();
partItem.loadAML(
  "<Item type='Part' action='add' >" +
    "<item_number>123-456</item_number>" +
    "<description>Blah blah</description>" +
    "<Relationships>" +
      "<Item type='Part BOM' action='add'>" +
        "<quantity>10</quantity>" +
        "<related_id>" +
          "<Item type='Part' action='get'>" +
            "<item_number>555-555</item_number>" +
```

```
          "</Item>" +
        "</related_id>" +
      "</Item>" +
    "</Relationships>" +
  "</Item>"
);

var resultItem = partItem.apply();
if (resultItem.isError()) {
  // handler error here
  return;
}
```

5.11 Add a Named Permission

You want to add a new named Permission Item.

Technique

Use the Item Class Extended Method set to add a new Named Permission Item.

C#

```
Item permItem = innovator.newItem("Permission", "add");
permItem.setProperty("name", "AK Part Permissions");

setIdentityAccess("All Employees", "get", true);
setIdentityAccess("CM", "get", true);
setIdentityAccess("CM", "update", true);
setIdentityAccess("CM", "delete", true);

Item resultItem = permItem.apply();
if (resultItem.isError())
{
    throw new Exception(resultItem.getErrorDetail());
}

void setIdentityAccess(string identityName, string permissionType, bool
accessState)
{
    Item identity = permItem.newItem();
    identity.setType("Identity");
    identity.setAction("get");
    identity.setProperty("name", identityName);

    Item access = permItem.newItem("Access", "add");
    access.setProperty("can_" + permissionType, accessState ? "1" : "0");
    access.setRelatedItem(identity);

    permItem.addRelationship(access);
}
```

5.12 Apply a Generic Method

You want to write Generic Methods that can be used as subroutines for other Methods.

Technique

Use the `Innovator.applyMethod()` method to apply Generic Methods. The following examples assume a server-side method named "Reverse String" exists, and that it returns a result item containing the reversed contents of the <string> tag.

C#

```
Innovator inn = this.getInnovator();
Item results = inn.applyMethod("Reverse String", "<string>abc</string>");
// Return a result item with "cba" as the contents of the Result tag
return inn.newResult(results.getResult());
```

5.13 Need to Send Email from a Method

You want to send an Email message from either a server or client-side Method.

Technique

Use the *Aras.IOM.Item.email(mail_item, idnt_item)* method to send email to a particular Innovator's identity.

> **Note:** Aras Innovator's identity could be a group of people in which case the email is sent to all of them.

C#

```
...
// It's assumed here that required identity (item of type 'Identity')
// has already obtained (see other examples on how to perform 'get'
// requests to Innovator server using IOM). Same about item of type
// 'User' that represents the person who sends the email ('fromUser').
Item idnt = ...
Item fromUser = ...

// It's assumed in the sample that this represents an item of
// type Part. ${Item/item_number} in the email message is a parameter
// that represents the XPath to the property item_number of
// item of type Part; this parameter will be substituted on
// this.item_number before the email is sent. Note that both
// subject and body of the email item could be parameterized.
// This mechanism allows to created parameterized email templates
// (items of type "Email Message") that could be saved
// in Innovator and used for sending emails with concrete content
// when required.
string subject = "Part promotion notification";
string body = @"The part ${Item/item_number} has been promoted";

// In this particular example instead of getting a ready template
// email from the server a new item of type "EMail Message" is created
Item email_msg = this.newItem("EMail Message");
```

```
email_msg.setProperty("subject", subject);
email_msg.setProperty("body_plain", body);
email_msg.setPropertyItem("from_user", fromUser);

// Finally send the email
if( this.email( email_msg, idnt ) == false )
{
  // Error handling
  ...
}
```

5.14 Want to Vault a File

You want to save a CAD Document with an attached Native File.

Technique

You will need to use the `setFileProperty` call which handles creating a file and sets the associated value on the specified property.

Note: For uploading large or complex data structures — including CAD items, images, drawings, and documents that involve File Items — it is strongly recommended to use the CheckinManager. CheckinManager is a utility class built into IOM, designed for structured and efficient batch uploads. It manages vaulting, parallel file transfer, event-driven progress reporting, and transaction handling for related items.

Note: You can also upload a file using a Stream by calling setFilePropertyViaStream() — useful when working with dynamic or in-memory file content. See the IOM API Reference for details on setFilePropertyViaStream.

C#

```
Item d = this.newItem("CAD", "add");
d.setProperty("item_number", "007");
d.setFileProperty("native_file", @"C:\myFile.txt");
var resultItem = d.apply();
if (resultItem.isError()) {
  // handler error here
  return;
}
```

5.15 Want to Download a Vaulted File

You want to download the file content associated with a File item or a property such as native_file on a CAD or Document item.

Technique

Use the fetchFileProperty() method to download the physical file associated with a File-type property on the item. The method handles locating the associated File item, accessing the vault, and saving the file to a specified location.

Note: When you need to download multiple files efficiently (e.g., from multiple items, or from large structures like assemblies), it is strongly recommended to use the CheckoutManager class. CheckoutManager supports parallel multi-threaded download, transactional handling, and event callbacks for progress and completion.

Note: If you want to download the file content as a stream — for example, to handle the file in memory or send it directly to another destination — use the fetchFilePropertyWithStream() method instead. This returns a Stream object from the vault server without saving the file locally.

C#

```
Item cadQry = myInnovator.newItem("CAD", "get");
cadQry.setProperty("item_number", "007");
cadQry.setAttribute("select", "id,native_file");

Item result = cadQry.apply();
if (result.isError())
{
   // handle error here
   return;
}

Item cad = result.getItemByIndex(0);
Item fileItem = cad.fetchFileProperty("native_file", @"C:\Temp\",
FetchFileMode.Normal);
string downloadedFilePath = fileItem.getProperty("checkedout_path");
```

5.16 How to Handle Multilingual Properties

You want to programmatically get/set multilingual string properties

Technique

Use *Item*.setAttribute("language","*") to get all language values and *Item*.setProperty(myProperty,value,lang) to set specific language values.

C#

```
// Add a new List with multilingual Value labels
var listItem = myInnovator.newItem("List","add");
listItem.setProperty("name","Numbers");
var valueItem = listItem.createRelationship("Value","add");
valueItem.setProperty("value","1");
valueItem.setProperty("label","One","en");
valueItem.setProperty("label","Ein","de");
var valueItem2 = listItem.createRelationship("Value","add");
valueItem2.setProperty("value","2");
valueItem2.setProperty("label","Two","en");
valueItem2.setProperty("label","Zwei","de");
var resultItem = listItem.apply();
if (resultItem.isError()) {
```

```
 // handle error here
 return;
}

// Retrieve the List with labels in both English and German
listItem = myInnovator.newItem("List","get");
listItem.setProperty("name","Numbers");
valueItem = listItem.createRelationship("Value","get");
valueItem.setAttribute("language","en,de");
resultItem = listItem.apply();
if (resultItem.isError()) {
  // handler error here
  return;
}
```

5.17 How to Handle Date Properties

You want to programmatically get/set date properties

Technique

Convert date values to "yyyy-MM-ddThh:mm:ss" format before setting the property

C#

```
// Get yesterday's date
DateTime myDate = DateTime.Today.AddDays(-1);

// Find all methods edited in the past 24 hours
Item myItem = myInnovator.newItem("Method","get");
myItem.setAttribute("select","name");
myItem.setProperty("modified_on",myDate.ToString("yyyy-MM-ddThh:mm:ss"));
myItem.setPropertyAttribute("modified_on","condition","gt");
myItem = myItem.apply();
```

6 Using CheckinManager

The CheckInManager is a function built into the IOM.dll which allows for the loading of large data structures into Aras Innovator that contain Images, Drawings, Documents, or other Items of the File Itemtype.
The following use cases describe how to use CheckInManager to submit images and drawings, either individually or as a collection.

6.1 Submitting Images

The following is the first step in submitting a File structure to CheckInManager to create a CAD item and add properties to it. The following code shows a typical AML query for creating a CAD item:

In this scenario, the setFileProperty method is required to add a property whose type is File. The parameters of setFileProperty include the Property's name and the fully qualified path of the File.

```
Item cad0 = innovator.newItem("CAD", "add");
cad0.setProperty("item_number", innovator.getNewID()); // required by CAD ItemType
cad0.setFileProperty("native_file", Path.Combine(directory, "input/native-0.cad"));
cad0.setFileProperty("thumbnail", Path.Combine(directory, "input/thumbnail-0.png"));
cad0.setPropertyAttribute("thumbnail", "checkinManager-type", "Image");
cad0.setProperty("name", "submitting-images-cad0");
```

If you need to use the CheckinManager to load an Image Property, you must set the Attribute *checkinManager-type* on the Property (such as in the thumbnail, below) to "Image". The following example shows the AML for the query:

```
<Item isNew="1" isTemp="1" type="CAD" action="add" id="FF678B0D9D43474E82305E232B5FF448">
        <item_number>57B354B0FEBF4F5BA5CCCCC6361A892B</item_number>
        <native_file>
                <Item isNew="1" isTemp="1" type="File" action="add" id="A276434AABD6420AA6640D0CAD4E97C3">
                        <filename>native-0.cad</filename>
                        <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\native-0.cad</actual_filename>
                        <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                        <Relationships>
                                <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                </Item>
                        </Relationships>
                </Item>
        </native_file>
        <thumbnail checkinManager-type="Image">
                <Item isNew="1" isTemp="1" type="File" action="add" id="ED6FF3D391E045C5A9D53424036B4862">
                        <filename>thumbnail-0.png</filename>
                        <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\thumbnail-0.png</actual_filename>
                        <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                        <Relationships>
                                <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                </Item>
                        </Relationships>
                </Item>
        </thumbnail>
        <name>submitting-images-cad0</name>
</Item>
```

If you are only importing one Item, you can then pass the Item directly to the CheckinManager. The following example uses `cad0` from the previous scenario:

```
using (CheckinManager manager = new IomFactory().CreateCheckinManager(cad0))
{
        Item response = manager.Checkin(numThreads);
        // …
}
```

You can also use this code to import more than one item. For example, given two CAD Items, referenced as `cad0` and `cad1` in the following example, you can generate 1 AML that contains both Items, as demonstrated here:

```
Item configuration = innovator.newItem();
configuration.loadAML("<AML>" + cad0.node.OuterXml + cad1.node.OuterXml + "</AML>");
```

The following is the AML code which would result from the `loadAML` statement:

```
<AML>
        <Item isNew="1" isTemp="1" type="CAD" action="add" id="FF678B0D9D43474E82305E232B5FF448">
                <item_number>57B354B0FEBF4F5BA5CCCCC6361A892B</item_number>
                <native_file>
                        <Item isNew="1" isTemp="1" type="File" action="add" id="A276434AABD6420AA6640D0CAD4E97C3">
                                <filename>native-0.cad</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\native-0.cad</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </native_file>
                <thumbnail checkinManager-type="Image">
                        <Item isNew="1" isTemp="1" type="File" action="add" id="ED6FF3D391E045C5A9D53424036B4862">
                                <filename>thumbnail-0.png</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\thumbnail-0.png</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </thumbnail>
                <name>submitting-images-cad0</name>
        </Item>
        <Item isNew="1" isTemp="1" type="CAD" action="add" id="4F2A8F67E6274E648AEDBD423637DBAE">
                <item_number>7120951EA16243D0BF28A0DDA32A0187</item_number>
                <native_file>
                        <Item isNew="1" isTemp="1" type="File" action="add" id="FE29E80F224F44EBA993E8FF54859DFE">
                                <filename>native-1.cad</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\native-1.cad</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </native_file>
                <thumbnail checkinManager-type="Image">
                        <Item isNew="1" isTemp="1" type="File" action="add" id="14367296E7ED45E284800F6F28127E3A">
                                <filename>thumbnail-1.png</filename>
                                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input\thumbnail-1.png</actual_filename>
                                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-images\bin\Debug\input</checkedout_path>
                                <Relationships>
                                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                                <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                        </Item>
                                </Relationships>
                        </Item>
                </thumbnail>
                <name>submitting-images-cad1</name>
        </Item>
</AML>
```

The following code can then be used to apply the query contained in the `configuration` value:

```
using (CheckinManager manager = new IomFactory().CreateCheckinManager(configuration))
{
        Item response = manager.Checkin(numThreads);
        // …
}
```

You must make sure you observe the following requirements for the query to work correctly:

- You must have at least one File or Image property otherwise CheckinManager will throw an exception.
- The Item passed to `CreateCheckinManager` cannot be null.
- A File item must have either an Add or a Create action associated with it.
- CAD items only support the following actions:
  - Add
  - Update
  - Delete
  - Version
  - Skip

  Specifying any other type of action results in an exception being thrown.

6.2 Submitting Drawings

In this use case, the Items are set up the same way they were set up in the previous use case. The difference is that the items created here have multiple parents. You must instantiate the MultiParentConfigurationBuilder class to add more than one parent. The following code relies on an item `configuration`, as shown in the previous examples:

```
MultiParentConfigurationBuilder builder =
factory.CreateMultiParentConfigurationBuilder(configuration);

Item drawingA = innovator.newItem("CAD", "add");
drawingA.setProperty("item_number", innovator.getNewID()); // required by CAD ItemType
drawingA.setFileProperty("native_file", Path.Combine(directory, "input/native-
A.cad"));
drawingA.setProperty("name", "submitting-images-drawingA");
builder.addParent(cadA.getID(), drawingA, "CAD Structure");
```

The following is the AML for the `cadA` Item from the example:

```
<Item isNew="1" isTemp="1" type="CAD" action="add" id="4C8DD7F5785E4E9D97461BDF2D1228D3">
        <item_number>7C9FD065FAAC4D4F955820E0FEC5CB5A</item_number>
        <name>submitting-drawings-cadA</name>
        <Relationships>
                <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                        <related_id>
                                <Item isNew="1" isTemp="1" type="CAD" action="add" id="774A4C99DDC7454E9D5A0BB84EA60459">
                                        <item_number>C8CB6EB081A24945A100C3CE8FE0BFA6</item_number>
                                        <name>submitting-drawings-cadB</name>
                                </Item>
                        </related_id>
                </Item>
                <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                        <related_id>
                                <Item isNew="1" isTemp="1" type="CAD" action="add" id="B236FBFD63CA4BD89AEF8A443E5942FE">
                                        <item_number>DCCA69BB52AE4CC185FBD3B450EF314B</item_number>
                                        <name>submitting-drawings-cadC</name>
                                </Item>
                        </related_id>
                </Item>
        </Relationships>
</Item>
```

The following shows the AML code for the drawingA Item:

```
<Item isNew="1" isTemp="1" type="CAD" action="add" id="AF1763AA429D41119C896E5170D1A725">
        <item_number>4816709F699946B595B4FBDF55FCE313</item_number>
        <native_file>
                <Item isNew="1" isTemp="1" type="File" action="add" id="8C83A9DD1EED4AA586CF6A53E7D542B3">
                        <filename>native-A.cad</filename>
                        <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-drawings\bin\Debug\input\native-A.cad</actual_filename>
                        <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-drawings\bin\Debug\input</checkedout_path>
                        <Relationships>
                                <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                                </Item>
                        </Relationships>
                </Item>
        </native_file>
        <name>submitting-drawings-drawingA</name>
</Item>
```

The addParent function creates a link between these two structures that generates the complete structure, as shown in the final CAD structure shown here:

The following is the AML code for *multiParentConfiguration*:

```
<AML>
    <Item isNew="1" isTemp="1" type="CAD" action="add" id="4C8DD7F5785E4E9D97461BDF2D1228D3">
        <item_number>7C9FD065FAAC4D4F955820E0FEC5CB5A</item_number>
        <name>submitting-drawings-cadA</name>
        <Relationships>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>
                    <Item isNew="1" isTemp="1" type="CAD" action="add" id="774A4C99DDC7454E9D5A0BB84EA60459">
                        <item_number>C8CB6EB081A24945A100C3CE8FE0BFA6</item_number>
                        <name>submitting-drawings-cadB</name>
                    </Item>
                </related_id>
            </Item>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>
                    <Item isNew="1" isTemp="1" type="CAD" action="add" id="B236FBFD63CA4BD89AEF8A443E5942FE">
                        <item_number>DCCA69BB52AE4CC185FBD3B450EF314B</item_number>
                        <name>submitting-drawings-cadC</name>
                    </Item>
                </related_id>
            </Item>
        </Relationships>
    </Item>
    <Item isNew="1" isTemp="1" type="CAD" action="add" id="AF1763AA429D41119C896E5170D1A725">
        <item_number>4816709F699946B595B4FBDF55FCE313</item_number>
        <native_file>
            <Item isNew="1" isTemp="1" type="File" action="add" id="8C83A9DD1EED4AA586CF6A53E7D542B3">
                <filename>native-A.cad</filename>
                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input\native-A.cad</actual_filename>
                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input</checkedout_path>
                <Relationships>
                    <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                    </Item>
                </Relationships>
            </Item>
        </native_file>
        <name>submitting-drawings-drawingA</name>
        <Relationships>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>4C8DD7F5785E4E9D97461BDF2D1228D3</related_id>
            </Item>
        </Relationships>
    </Item>
    <Item isNew="1" isTemp="1" type="CAD" action="add" id="D0AD5B11F58F4CB79978C2FB4BFD1A71">
        <item_number>D54200E995B9410F808EEB26FB9E04AB</item_number>
        <native_file>
            <Item isNew="1" isTemp="1" type="File" action="add" id="DFE1AA07106B4AC4B63595E8CC89833F">
                <filename>native-B.cad</filename>
                <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input\native-B.cad</actual_filename>
                <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                    drawings\bin\Debug\input</checkedout_path>
                <Relationships>
                    <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                        <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                    </Item>
                </Relationships>
            </Item>
        </native_file>
        <name>submitting-drawings-drawingB</name>
        <Relationships>
            <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                <related_id>774A4C99DDC7454E9D5A0BB84EA60459</related_id>
```

```
                </Item>
            </Relationships>
        </Item>
        <Item isNew="1" isTemp="1" type="CAD" action="add" id="E5F643DCC71A4740AC29189B064C9AC6">
            <item_number>5FE746744D8C433281F32C9588F2D64B</item_number>
            <native_file>
                <Item isNew="1" isTemp="1" type="File" action="add" id="8C4C44D934B14CE0A8779008F4395F89">
                    <filename>native-C.cad</filename>
                    <actual_filename>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
                        drawings\bin\Debug\input\native-C.cad</actual_filename>
                    <checkedout_path>C:\Sandbox\pbellis\checkinmanagerexamples\CompilableCode\examples\submitting-
drawings\bin\Debug\input</checkedout_path>
                    <Relationships>
                        <Item type="Located" action="add" where="related_id='67BBB9204FE84A8981ED8313049BA06C'">
                            <related_id>67BBB9204FE84A8981ED8313049BA06C</related_id>
                        </Item>
                    </Relationships>
                </Item>
            </native_file>
            <name>submitting-drawings-drawingC</name>
            <Relationships>
                <Item isNew="1" isTemp="1" type="CAD Structure" action="add">
                    <related_id>B236FBFD63CA4BD89AEF8A443E5942FE</related_id>
                </Item>
            </Relationships>
        </Item>
</AML>
```

In this case, `cadA` is equal to *configuration*. The last line of code creates a `CAD Structure` relationship between `drawingA` and `cadA`. The example repeats similar logic for `cadB` and `cadC`. The following code then applies the query:

```
Item multiParentConfiguration = builder.GetItemConfiguration();

using (CheckinManager manager =
factory.CreateCheckinManager(multiParentConfiguration))
{
        Item response = manager.Checkin(numThreads);

        // …

}
```

The first line converts the Builder's query into a query that supports multiple parents as determined by the calls to *builder.addParent*. After the first line of code, query execution is applied in the same way as a standard CheckInManager query.

6.3 Async Processing

The previous use cases allow the main thread to do the processing. In this use case, the class offloads all logic to a separate thread from the executing thread. Because of this, you need to keep the following in mind:

- Using the Dispose pattern as the main thread continues execution and disposes of the `CheckInManager` will cause a `NullReferenceException`.
- Retrieving the results from the `CheckinManager` requires a callback for either `UploadFilesCompleted` or `CheckinCompleted`. `UploadFilesCompleted` loops through all the files and writes out which files have been added. If an error is encountered, an exception is thrown. `CheckInCompleted` stores a list of all the results.
- In `CheckinCompleted`, you must dispose of the `CheckInManager` to prevent a resource leak.

7 Using CheckoutManager

The CheckoutManager is a class built into IOM.dll that enables efficient downloading of large sets of files stored in Aras Innovator's vault. It is designed for scenarios where your application needs to retrieve multiple files (e.g., from CAD structures, documents, or drawings) and provides a batch-oriented, multi-threaded, and optionally transactional mechanism for doing so.

The following use cases describe how to use CheckoutManager to perform efficient file downloads, whether for single or multiple items.

7.1 Downloading Files from an Item Structure

The example below demonstrates how to retrieve a CAD Item and its related CAD Structure, and download all associated native files in one operation using CheckoutManager.

C#

```
string cadId = "you_cad_id";

string amlQuery =
$@"<AML>
  <Item type=""CAD"" select=""item_number,native_file(filename)""
action=""GetItemRepeatConfig"" id=""{cadId}"">
    <Relationships>
      <Item type=""CAD Structure"" select=""related_id"" repeatTimes=""0""
repeatProp=""related_id"">
        <related_id>
          <Item type=""CAD"" />
        </related_id>
      </Item>
    </Relationships>
  </Item>
</AML>";

Item result = myInnovator.applyAML(amlQuery);

if (result.isError())
{
   throw new Exception("Error retrieving CAD and its Structure: " +
result.getErrorString());
}

using (var checkoutManager = new CheckoutManager(result,
CheckoutManagerFlags.UseTransactions))
{
   Hashtable checkoutResult = checkoutManager.DownloadFiles(@"C:\Temp\", 4);
   foreach (string fileId in checkoutResult.Keys)
   {
      var downloadResult = checkoutResult[fileId] as DownloadResult;

      if (downloadResult.Error != null)
      {
```

```
         Console.WriteLine($"Error downloading file: {fileId} -
{downloadResult.Error.Message}");
      }
      else
      {
         var downloadedFilePath =
downloadResult.Result.getProperty("checkedout_path");
         Console.WriteLine($"Downloaded file: {fileId} -
{downloadedFilePath}");
      }
   }
}
```

COM

```
IOM::IItemComIncomingPtr result = myInnovator->applyAML(aml);

IOM::ICheckoutManagerComIncomingPtr checkoutManager = iomFactory-
>CreateCheckoutManager(result, IOM::CheckoutManagerFlags_UseTransactions);

mscorlib::IDictionaryPtr downloadResults = checkoutManager-
>DownloadFiles(targetDir, 4);

IOM::IItemComIncomingPtr files = result-
>getItemsByXPath("//Item[@type='File']");

for (int fileIndex = 0; fileIndex < files->getItemCount(); fileIndex++)
{
   IOM::IItemComIncomingPtr file = files->getItemByIndex(fileIndex);
   IOM::IDownloadResultPtr downloadFileResult = downloadResults->Item[file-
>getID()];

   IOM::IItemComIncomingPtr resultItem = downloadFileResult->Result;
}
```

8 Appendix A: Local NuGet Repository

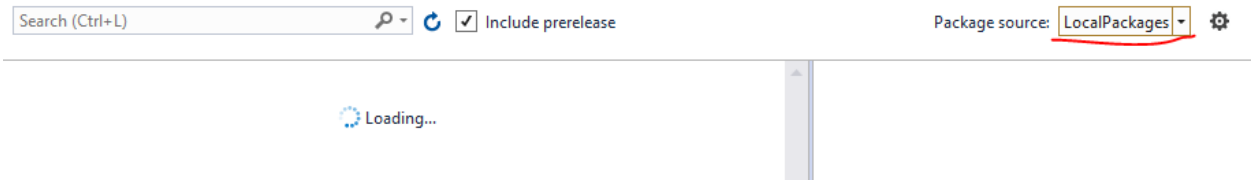
If nuget.org is not accessible, you can still consume the Aras IOM SDK packages by creating a local NuGet feed.

One of possible NuGet feeds is a local NuGet repository. To create a local repository, you need to do following steps:

1. Create LocalPackages directory in repository of your project
2. Copy NuGet packages (e.g., Aras.IOM.*.nupkg) to the LocalPackages folder
3. Create NuGet.Config file in root directory of your project
4. Put following value in Nuget.Config file

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <packageSources>
    <add key="localPackages" value="LocalPackages" />
  </packageSources>
</configuration>
```

5. Run Visual Studio
6. Open the project in which you need to reference NuGet-package
7. Perform right mouse click on project in solution explorer and click 'Manage NuGet packages' in context menu
8. Choose LocalPackages in Package source



9. Search for necessary package and install.

Note: You can find more information about private NuGet feeds on https://docs.microsoft.com/en-us/nuget/hosting-packages/overview