Getting Started Developer Guide

iAnywhere® Mobile Office 5.7

SYBASE® iAnywhere

Document ID: DC01170-01-0570-01
LAST REVISED: *September 2009*

Copyright and Trademarks

Disclaimer

This documentation, as well as the software described in it, is furnished under a license agreement. The content of this documentation is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by iAnywhere Solutions, Inc. iAnywhere Solutions, Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in this documentation. Any changes in the programs will be incorporated in a future edition of this publication.

Technical Support

For product-specific technical information, visit the iAnywhere Technical Support site at http://frontline.sybase.com/support/.

NOTE: Register at our technical support site for the latest information for your product. This site is available only to customers with a valid maintenance contract.

United States
+1 800 235 7576, menu options 2, 1
+1 208 322 7575, menu options 2, 1
6:00 a.m. to 6:00 p.m. Mountain Time (GMT -7)

United Kingdom
+44 (0) 117 333 9032
8:00 a.m. to 6:00 p.m. (GMT+1)

Germany
+49 (0) 7032 798-555
8:00 a.m. to 6:00 p.m. (GMT+1)

France
+33 (0) 825 826 835
8:00 a.m. to 6:00 p.m. (GMT+1)

Benelux
+31 (0) 302 478 455
8:00 a.m. to 6:00 p.m. (GMT+1)

Table of Contents

Welcome

Welcome to iAnywhere® Mobile Office, a component of the Sybase® Information Anywhere® Suite. iAnywhere Mobile Office is specifically designed for today's mobile business workforce. It combines fully integrated wireless email and PIM with on-device security and business process mobilization. iAnywhere Mobile Office enables organizations to manage critical, time-sensitive workflow business processes. iAnywhere Mobile Office offers key features that provide the foundation for a company's mobile inbox of the future.

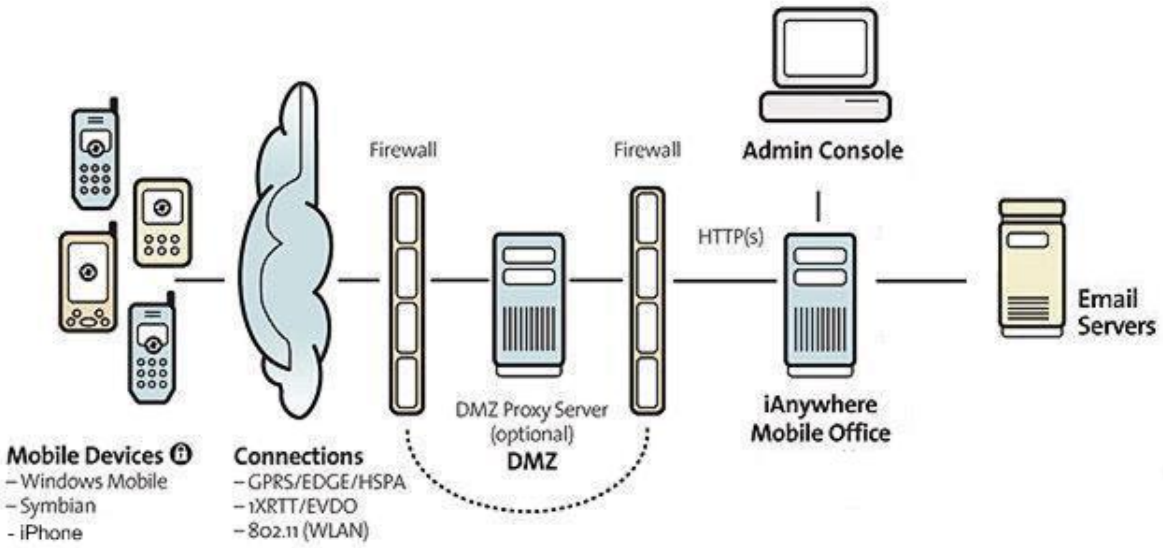
The unique iAnywhere Mobile Office SDK enables enterprises to extend their business processes to mobile workers in a secure and reliable way. Enterprise backend services implemented as Web Services or developed using the .NET Framework technology can be easily made available to mobile applications. There is no need to worry about the physical network, data encryption or authentication from the mobile device, enterprise developers can focus on the application logic itself to provide the best user experience possible with today's mobile application development tools like Visual Studio and the .NET Compact Framework.

See Also:

Workflow Developer Guide in Start > Programs > iAnywhere > Mobile Office Docs.

Architecture Overview

The image below shows the architecture for iAnywhere Mobile Office.



Mobile Office clients communicate with the server infrastructure using a secure optimized protocol for today's always-on wireless networks of all major carriers worldwide.

The DMZ Proxy adds an extra layer of security. This server component deployed inside the enterprise DMZ allows secure behind-the-firewall deployment of the iAnywhere Mobile Office server. The iAnywhere Mobile Office server makes outbound connections into the enterprise DMZ using the secure HTTPS protocol. When using this component, connections from all mobile devices end within the DMZ of your enterprise.

The iAnywhere Mobile Office server communicates with connected enterprise back-end applications using their native protocols. No data is staged or stored within the enterprise DMZ at any time.

Integration modules for enterprise back-end applications are implemented as .NET Assemblies or COM Objects installed on the iAnywhere Mobile Office Server. When configured in the service registry of the iAnywhere Mobile Office server all services are transparently available to mobile applications.

Mobile Office SDK

The Mobile Office SDK is based on Mobile Objects technology, a technology that supports a distributed objects framework. Objects or services available within an enterprise network can be made available to mobile applications in an easy and transparent way for the developer. The most popular example of a real-world usage scenario for the Mobile Office SDK is the iAnywhere Corporate Directory application, which allows users to browse the enterprise address book. Detailed information about the available features and functions can be found in the Mobile Objects chapter of the documentation on the installation CD image (.*Docs**OBMG**MobileDataSuite.chm*).

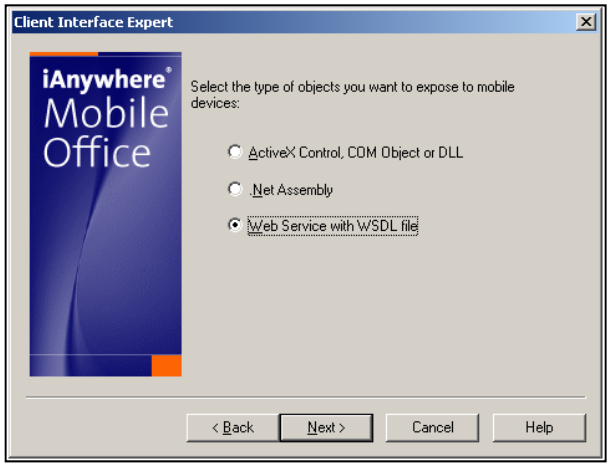
Mobilizing an enterprise service with the Mobile Office SDK is a simple three-step process:

Step 1 – Identifying the enterprise service and generating the source code to interact with the service from within a mobile application.

Step 2 – Enabling the enterprise service within the Service Registry to allow access from the mobile application.

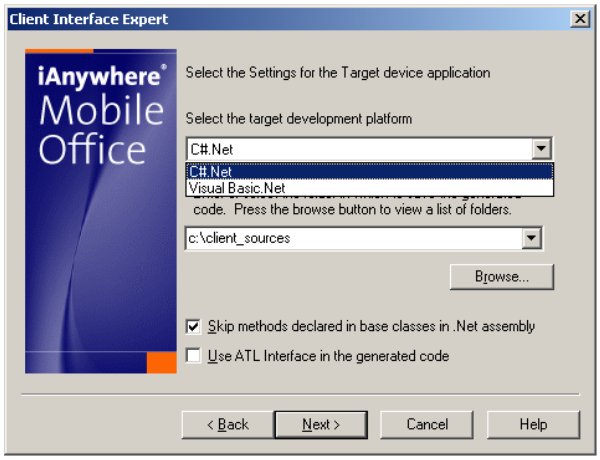
Step 3 – Creating the mobile application to use the service.

Client Interface Expert (CIE)



The Client Interface Expert is used to generate the necessary source code that needs to be compiled into the mobile application to utilize services registered on the iAnywhere Mobile Office server.

This utility can be found in the iAnywhere Mobile Office Server installation directory (.*Tools**MO_SDK**CIE*).



Mobile Object Method Security (MOMS)

Access control to services accessible via Mobile Objects is implemented as an object registry that allows only certain devices access to certain methods. This enables a base level separation between normal users and administrative users. The object registry provides a concept of method grouping and the ability to assign such method groups to individuals devices who can then be granted execution rights to specific methods contained in a group via Mobile Objects. This object registry replaces the existing <AvailableServices> section in the MOAdapterSettings.xml.

Configuring MOMS

If you are writing and deploying custom Mobile Objects applications using the MO SDK, you would do the following to configure MOMS.

Add Server MO Objects to the MOMS Configuration Tables

When you deploy your application, you will need to add your server Mobile Objects methods to the MOMS configuration tables. Otherwise, your client application will not be able to invoke your server methods. Edit the MOMS configuration tables in the database using a tool such as Advantage Data Architect (ARC).

The configuration tables are located at C:\Program Files\Extended Systems\OneBridge Sync Server\Data\OBR.

Note: Do not remove any rows that are installed by Mobile Office as these rows are needed for the product to run correctly.

The tables are as follows:

The MOMS_ROLE table will contain a list of supported roles:

Column	Type	Description
ROLE_ID	integer	Internal Role ID (PK).
ROLE_NAME	char(100)	Name of the role. Used for documentation only.

The MOMS_DEVICE_ROLE table associates devices with roles. Note that a device could be assigned to multiple roles, in which case the device's white list is a union of the permissions of all the roles to which it is assigned. Most devices will be assigned a single role.

Column	Type	Description
DEVICE_ID	integer	Internal Device ID (PK, FK)
ROLE_ID	integer	Internal Role ID (PK, FK)

The MOMS_FUNCTIONAL_AREA table will contain a list of supported functional areas. A functional area is a grouping of object/method names:

Column	Type	Description
FUNCTIONAL_AREA_ID	integer	Internal FA ID (PK).
FUNCTIONAL_AREA _NAME	char(100)	Name of the functional area. Used for documentation only.

The MOMS_FUNCTIONAL_AREA_METHOD table enumerates the object/method combinations that are part of a functional area.

Column	Type	Description
FUNCTIONAL_AREA_ID	integer	Internal Functional Area ID (PK, FK To MOMS_FUNCTIONAL_AREA table).
OBJECT_NAME	char(150)	Mobile Objects method name that is part of this functional area. (PK). Names will be stored in all-uppercase to facilitate case-insensitive comparisons.
METHOD_NAME	char(50)	Mobile Objects method name that is part of this functional area. (PK). Names will be stored in all-uppercase to facilitate case-insensitive comparisons.

The MOMS_ROLE_FUNCTIONAL_AREA table associates roles with functional areas. Note that a role could be assigned to multiple functional areas, in which case the role's white list is a union of the permissions of all the functional areas to which it is assigned.

Column	Type	Description
ROLE_ID	integer	Internal Role ID (PK, FK)
FUNCTIONAL_AREA_ID	integer	Internal FA ID (PK, FK)

MOMS Role Assignments

When a user is registered, it will be associated with an appropriate pre-existing role in the Robie database. For the initial release, the pre-existing roles will be the following:

- Admin (provisioned remote Admin app users)
- Server Addin (current "ServerComponent" processes, like local Admin, OBAgent, and OBListener).
- Device User (currently any user registered through Admin)

The user role assignment is done implicitly in the code at the time of user registration, based on the context of the registration.

Client library

The Mobile Objects client library needs to be referenced by all mobile applications that intend to utilize services exposed through the iAnywhere Mobile Office Server. It contains the core functionality to establish a secure and optimized connection from a mobile application to the iAnywhere Mobile Office environment. The client library is available for all supported Windows Mobile device platforms. This library can be found in the installation directory of the iAnywhere Mobile Office Server (.*Tools**MO_SDK**dotNet*). The client library requires a native runtime on the device that is normally installed together with the Mobile Office Client components for PIM and E-Mail access.

Note: Currently a number of methods and classes contained in the client library are unsupported with iAnywhere Mobile Office. For example, these are the DoAction, GetResults, and all push-related methods of the MOConnection class, the MOActionNode class, the MOLiveConnectListener class and the MOProfileStore class. Functionality provided via these methods and classes is currently only available with the OneBridge Mobile Data Suite product.

Hello World Sample

Server-side Object

```
public class CHelloWorld
{
   private string m_remoteuser = "";
   private string m_remotedevice = "";

   public void MORecvUserInfo(DataTable dtUserInfo)
   {
      foreach (DataRow drUserInfoEntry in dtUserInfo.Rows)
      {
         String sValueName = null, sValueData = null;

         sValueName = Convert.IsDBNull(drUserInfoEntry["VALUE_NAME"])
            ? null : Convert.ToString(drUserInfoEntry["VALUE_NAME"]);

         if (!String.IsNullOrEmpty(sValueName))
         {

            sValueData = Convert.IsDBNull(drUserInfoEntry["VALUE_DATA"])
               ? null : Convert.ToString(drUserInfoEntry["VALUE_DATA"]);

            switch (sValueName)
            {
               case "USER_ID":
                  m_remoteuser = sValueData;
                  break;
               case "DEVICE_ID":
                  m_remotedevice = sValueData;
                  break;
            }
         }
      }
   }

   public string HelloWorld()
   {
      return "Hello " + m_remoteuser + "!";
   }
}
```

Client-side Application

```
static void Main()
{
   MOConnection moconn = new MOConnection();
   MORequestOptions moreqop = new MORequestOptions();

   moconn.Init("moserver", // the servername
            5001, // the serverport
            "CID", // the company id
            "user", // the username
            false, // currently unused set to false
            "", // currently unused set to empty string
            true, // should the comunication be encrypted
            "code", // the assigned activation code
            "OBPIM", // currently unused set to "OBPIM"
            "C1"); // the unique connection id

   CHelloWorld sample = new CHelloWorld(moconn);
   string result = sample.HelloWorld(moreqop);
   moconn.Close();

   Console.WriteLine(result);
}

public class CHelloWorld : MOObject
{

   public CHelloWorld( MOConnection conn ) : base( conn )
   {
      ProxyVersion = 1;
   }

   public string HelloWorld( MORequestOptions requestOptions )
   {
      ClearParams();
      Execute( requestOptions,
              "monet:HelloWorldSample.dll:HelloWorldSample.CHelloWorld",
               "HelloWorld" );

      if ( requestOptions.Command == enumCommand.cmdInvokeWait ||
           requestOptions.Command == enumCommand.cmdRetrieve )
      {
         MOParam param;
         param = GetReturnParam();
         return param.GetString();
      }
      else
      {
         return null;
      }
   }
}
```