



Magnet™ iOS Developer Guide

2.0

Revision A

© 2013 Magnet Systems, Inc. All rights reserved. This manual, in whole or in part, may not be reproduced, translated, or reduced to any machine-readable form without the prior written approval of Magnet Systems, Inc. Magnet Systems, Inc. reserves the right to make any modification to this manual or the information contained herein at any time without notice.

MAGNET SYSTEMS PROVIDES NO WARRANTY WITH REGARD TO THIS MANUAL, THE SOFTWARE, OR OTHER INFORMATION CONTAINED HEREIN AND HEREBY EXPRESSLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE WITH REGARD TO THIS MANUAL, THE SOFTWARE, OR SUCH OTHER INFORMATION. IN NO EVENT SHALL MAGNET SYSTEMS, INC. BE LIABLE FOR ANY INCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, WHETHER BASED ON TORT, CONTRACT, OR OTHERWISE, ARISING OUT OF OR IN CONNECTION WITH THIS MANUAL, THE SOFTWARE, OR OTHER INFORMATION CONTAINED HEREIN OR THE USE THEREOF.

All other trademarks are trademarks of their respective owners. Any Magnet Systems patents granted or pending in the United States and other countries are protected by law.

Magnet Systems, Inc.

435 Tasso Street
Suite 100
Palo Alto, CA
94301

650-329-5904

info@magnet.com

1. About This Guide	1
Audience	1
Document Organization	1
Related Documents	2
2. Introducing Magnet Mobile App Server.....	3
Magnet Mobile Enterprise Server	4
Custom Controllers	4
Caching and Off-line Messaging	4
Persistence	4
Transactions	5
Service Integration	5
Third-Party Service Controllers	5
Magnet Mobile App Manager	6
Mobile App Management Console	6
Mobile App Store	6
Deployment	7
Developer Factory	7
3. Getting Started	9
Set Up Your IDE Environment	9
Set up Your Account in Magnet Developer Factory	9
4. Managing Your Project with Developer Factory	11
Creating a Project	12
Displaying the Generated Assets	19
Modifying a Project	20
Uploading a Project	22
Removing a Project	23
Inviting Other Developers	25

5. Defining Log In and Out Services	27
loginWithCredentials	28
loginWithSavedCredentials	29
Logout	29
6. Defining Caching and Off-line Mode Services.....	31
Declaring Options	32
Creating MMReliableCallOptions	32
Setting Cache Age	33
Setting Request Timeout	33
Creating MMAsyncCallOptions	33
Setting Cache Age	34
Setting Constraints	34
Making a Reliable Request	35
Caching Results and Callback	35
Caching Results but no Callback	35
No Caching and No Callback	35
Making an Asynchronous Request	36
Caching Results and Callback but no Reliability	36
No Caching, Reliability, or Callback	36
No Caching or Reliability but With Callback	36
Applying a Constraint to an Asynchronous Request	37
7. Using Third-Party Social Controllers	39
Using a Salesforce Controller	39
Using a Facebook Controller	40
8. Invoking a Controller	41
Sending a Message to a Controller	41
Using Node	42
Using MMData	43
9. Assembling Your Project	45
10. Showing TechCare Sample App	47



The Magnet™ iOS Developer Guide provides information and code samples for building enterprise native iOS apps. It also provides instructions for uploading the generated project to a private cloud-based sandbox in the Magnet Developer Factory for testing.

Audience

This guide is intended for developers who write iOS client-side apps that interact with the Magnet Enterprise Server applications. This document assumes that you are experienced in an iOS development environment, such as Xcode and Objective-C frameworks.

Document Organization

This guide includes these chapters:

- *Introducing Magnet Mobile App Server*, provides a high-level description of each component, its relationship, and interaction between the components.
- *Getting Started*, provides information that you need to gather before writing your app.
- *Managing Your Project with Developer Factory*, provides instructions for creating, modifying, and deleting your project by way of Magnet Developer Factory.
- *Defining Log In and Out Services*, provides methods used in user log in and log out services.
- *Defining Caching and Off-line Mode Services*, provides methods used for making reliable and asynchronous requests, and code samples for each request scenario.
- *Using Third-Party Social Controllers*, provides sample code for using Salesforce and Facebook controllers.

- *Invoking a Controller*, provides construct to send a message to a controller. It also provides code samples for MMData object and node.
- *Assembling Your Project*, provides information for compiling your project.
- *Showing Sample Code*, shows use cases and sample code used in the TechCare app.

Related Documents

The following table lists and describes other documents that are related to the Magnet products.

Document Title	Description
Magnet™ Server Application Developer Guide	Intended for developers who write Java-based server-side application, this guide provides information and code samples for building Magnet Enterprise Server applications.
Magnet™ Android Developer Guide	Intended for Android app developers, this guide provides information and code samples for building enterprise native Android apps. It also provides instructions to upload the completed Android app to your private sandbox for testing.
Magnet™ Mobile App Server Deployment Guide	Intended for IT administrators, this guide provides information and instructions for deploying the Magnet Mobile App Server to the Amazon cloud.
Magnet™ Mobile App Administrator Guide	Intended for IT administrators, this guide provides information and instructions for managing apps using the Magnet Mobile App Management Console.
Magnet™ Mobile for Android User Guide	Intended for users of Android devices, this guide provides information and instructions for installing the Magnet Mobile Server and using Apps@Work.
Magnet™ Mobile for iOS User Guide	Intended for users of iOS devices, this guide provides information and instructions for installing Setup@Work and using Apps@Work.

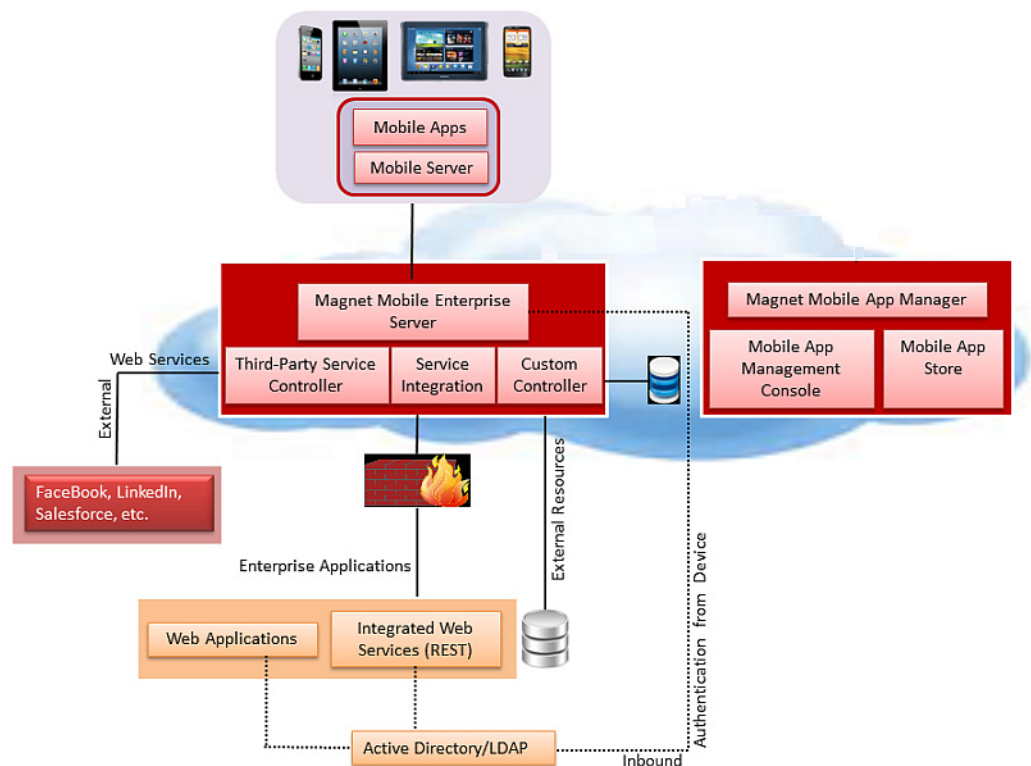


2. Introducing Magnet Mobile App Server

The Magnet™ Mobile App Server provides basic constructs for creating and manipulating app objects spanning the server and mobile app contexts. The Magnet Mobile App Server:

- Enables data flow between users' devices and enterprise back-end systems, as well as monitoring usage of mobile apps.
- Supports performance, availability, scalability, and reporting.
- Ensures that security-sensitive data is encrypted locally on device storage.

The following diagram shows the Magnet Mobile App Server components and their relationships.



Magnet Mobile Enterprise Server

Magnet Mobile Enterprise Server (MES) is a server that contains all business logic components in a single deployable JAR file. Each encapsulation of business logic is herein referred to as a Controller. Instances of the Magnet Mobile Enterprise Server are deployed in the cloud, and may contain a blend of off-the-shelf (OTS) and custom controllers that you write for your app.

Custom Controllers

Should you want to create your own custom controllers, you can download the source code for a reference server controller implementation. You can then modify this reference controller implementation to add your own business logic, then upload the modified project for your custom controller back into the Magnet Developer Factory so that this custom controller can be included into a build of the MES instance along with other controllers defined in your project. Custom controllers leverage services such as caching and off-line mode and persistence.

Caching and Off-line Message Delivery

To deal with unreliable connections between mobile devices and MES and to ensure reliable message delivery, Magnet Mobile App Server supports off-line operations that store and cache messages, and forwards information when the network connection is reestablished.

You can invoke these asynchronous services locally on the server or remotely from a mobile client. The invocations from the mobile client can occur asynchronously and reliably (reliability is an option). They are asynchronous in that the invocation is recorded for later playback to the server. They are reliable in that after an invocation has been submitted, it is guaranteed to be run once—and only once from the server perspective.

After entries are processed and results arrive back from the server, a listener that you optionally provide will be notified. Listeners are either transient or durable. A transient listener will be forgotten if the app JVM instance is cycled for any reason. Durable (non-transient) listeners will survive JVM restarts.

Persistence

Persistence is about the storage and retrieval of structured data. Magnet Mobile App Server provides support for entities and mapping entities to the data store. This creates, in effect, a “virtual entity data store” over any combination of MySQL, LDAP, and so forth, that can be used from within the programming interface Magnet provides.



Magnet Mobile App Server uses Java interfaces to define functions, methods, classes, and requests. Magnet Mobile App Server generates implementation from these classes built on annotations declared on interfaces, which allows for relationship, attributes, and so forth. Using user-friendly interfaces on entities, the Magnet Mobile App Server provides these features:

- Queries on entity with query composition
- CRUD (Create, Read, Update, Delete) operations on entities
- Complex queries such as paging and ordering

A developer can choose any technology (JDBC, Hibernate, etc.) to access a data store. However, implementing with Magnet persistence APIs is easier to use, and powerful over time. Every time an entity is used, unstructured data is collected. Over time, recommendations can be derived from that unstructured data. For example, when something is changed on an employee record, the implementation will track when that change is made and by whom. Of course, the recommendations will only be provided if you choose to incorporate those modules into your app. The entire methodology of building apps is modular - allowing you to blend the right combination of modules that make sense for your app.

Transactions

Magnet Mobile App Server can optionally use transactions to maintain the integrity of data.

Service Integration

Service integration can automatically transform Web applications from legacy enterprise application servers to the Magnet Mobile Enterprise Server. It exposes traditional SOAP-based Web Services hosted in the enterprise application servers to REST-based services for mobile access. Using the free on-line Developer Factory, you create a project with specific requirements, and download the generated APIs that abstract SOAP/REST services as controllers. Once generated, you can customize them as you see fit.

Third-Party Service Controllers

The third-party service controllers are sample controllers that allow you to connect to well-known social services, such as Facebook, LinkedIn, or Salesforce. These controllers have built-in support for OAuth, so you can securely retrieve contacts information, and share an update. The source code for these controllers is included in the project generated by the Developer Factory if you choose to include them in your project.



Magnet Mobile App Manager

The Magnet™ Mobile App Manager (MAM) is a run-time server that monitors the service status of Magnet MES and provides app management functionality by way of the Magnet Mobile App Management Console and Mobile App Store.

Mobile App Management Console

The Magnet MAM Console is a Web interface of the Magnet Mobile App Manager that enables IT administrators to centrally manage and administer apps on client devices. Access to the MAM Console is defined by role-based and group association, which is defined in LDAP. Using the MAM Console, IT administrators can:

- Enable employee BYOD.
- Remotely install, remove, upgrade, and track apps.
- Customize an enterprise private-label app store.
- Remotely activate / deactivate user accounts.
- Push required apps to authorized users.
- View logs to monitor events history.

Mobile App Store

The Mobile App Store is an app for installing and managing apps on a mobile device and is installed on iOS and Android devices. The Mobile App Store includes a suite of sample apps that are organized in categories. Via the MAM Console, IT administrators can brand the Mobile App Store with the enterprise private label. The App Store branding kit can be obtained on request.

Deployment

An executable JAR file contains all controllers required for a specific deployment. Instances of the Magnet Mobile Enterprise Server are deployed in the Amazon cloud—a virtual private cloud that is managed by customers.



Developer Factory

The Developer Factory is a free on-line service that provides a secure and private place for you to dynamically construct controller interfaces specific to the back-end services required for use in your mobile apps. The Developer Factory empowers you to

- Create and manage your app projects.
- Generate customized controllers and runtime binaries for your projects.
- Obtain source code for all your customized project assets.
- Upload modified controller and entities source codes for custom app controllers to regenerate executable JAR files.
- Obtain binaries for additional components required to deploy your projects in the Amazon cloud. Automatically create other language packs of your controllers (for example, iOS/XCode).
- Test-drive your app projects by getting access to your own private sandbox environment in the cloud.





Native iOS apps are built using the iOS system frameworks and Objective-C language. Native apps are installed physically on the device and can run without the presence of a network connection.

Setting Up Your IDE Environment

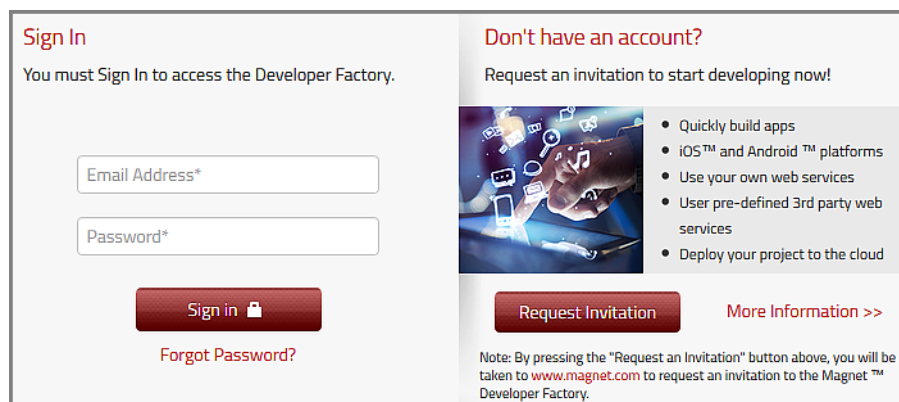
To develop applications for iOS, you need a Macintosh computer, iOS software development kit (SDK), and Xcode. Xcode is the development environment you use to create, test, debug, and tune your applications.

Setting up Your Account in Magnet Developer Factory

To log in to the Magnet Developer Factory:

- 1 Launch a browser, and enter the following URL:

<http://factory.magnet.com>



The screenshot shows the Magnet Developer Factory login and registration interface. On the left, under the heading "Sign In", there is a message "You must Sign In to access the Developer Factory." Below this are two input fields: "Email Address*" and "Password*". A red "Sign in" button with a lock icon is positioned below the password field, and a red link "Forgot Password?" is at the bottom. On the right, under the heading "Don't have an account?", there is a message "Request an invitation to start developing now!". Below this is a graphic of a hand holding a smartphone with various app icons floating around it. To the right of the graphic is a list of benefits: "Quickly build apps", "iOS™ and Android™ platforms", "Use your own web services", "User pre-defined 3rd party web services", and "Deploy your project to the cloud". Below the graphic is a red "Request Invitation" button and a red link "More Information >>". At the bottom right, a note states: "Note: By pressing the 'Request an Invitation' button above, you will be taken to [www.magnet.com](\"http://www.magnet.com\") to request an invitation to the Magnet™ Developer Factory."

- 2 Click **Request Invitation** to start the process.



4. Managing Your Project with Developer Factory

The Developer Factory provides an intuitive environment enabling you to easily build and manage your project. As an owner of a project that you create, you can:

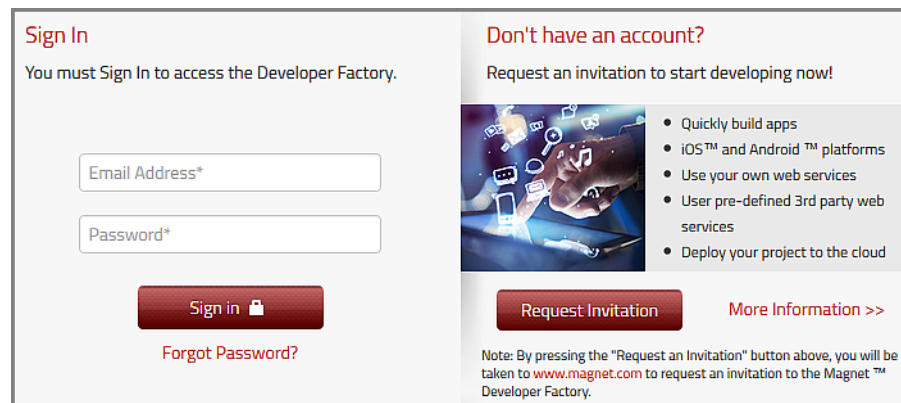
- Create a project (page [12](#))
- Deploy your project for testing or download it for customization (page [19](#))
- Modify your project (page [20](#))
- Upload a customized project (page [22](#))
- Remove your project (page [23](#))
- Invite other developers (page [25](#))

Creating a Project

The Magnet Developer Factory provides wizard that walks you through the creation of a project.

- 1 Launch a browser and enter the following URL:

<http://factory.magnet.com>

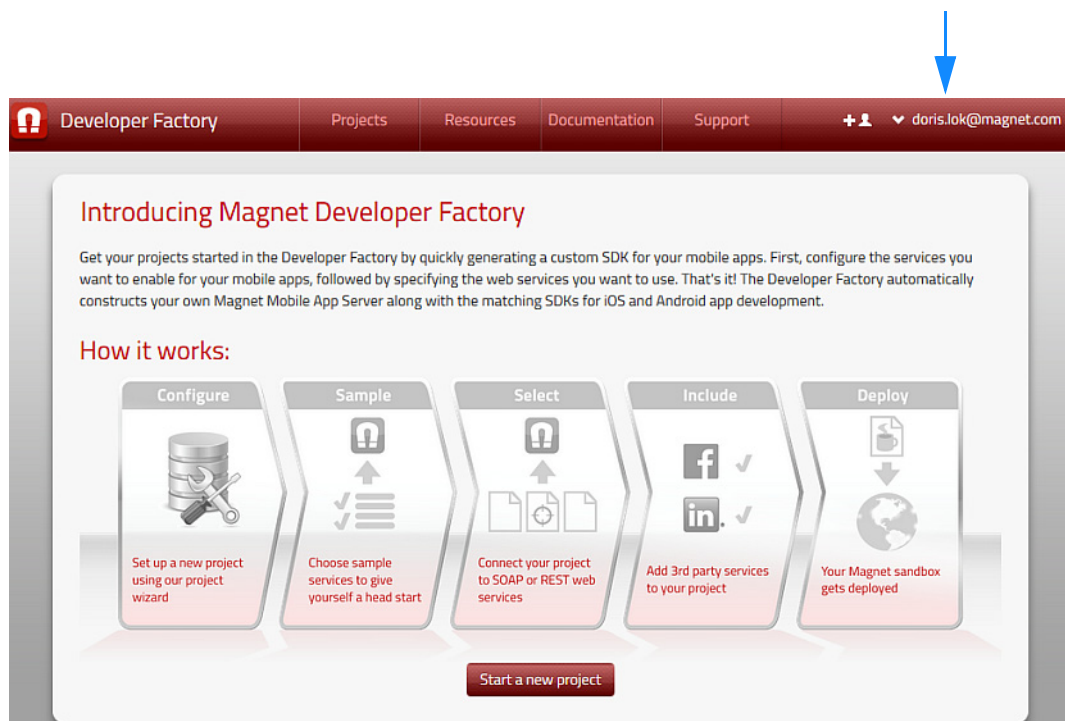


The screenshot shows the login interface of the Magnet Developer Factory. On the left, under the heading "Sign In", there is a message "You must Sign In to access the Developer Factory." Below this are two input fields: "Email Address*" and "Password*", followed by a "Sign in" button and a "Forgot Password?" link. On the right, under the heading "Don't have an account?", there is a message "Request an invitation to start developing now!". Below this is a "Request Invitation" button and a "More Information >>" link. A list of features is shown: "Quickly build apps", "iOS™ and Android™ platforms", "Use your own web services", "User pre-defined 3rd party web services", and "Deploy your project to the cloud". A note at the bottom right states: "Note: By pressing the 'Request an Invitation' button above, you will be taken to [www.magnet.com](\"http://www.magnet.com\") to request an invitation to the Magnet™ Developer Factory."

- 2 Enter your user credentials information in the appropriate fields:

- 3 Click **Sign in**.

The page with your user name displayed opens, as shown in the following example.



4 Click **Start a new project**.

5 Enter the name of your project, version number, and a brief description in the appropriate fields.

6 Click **Next Step**.

The name of the project you created is displayed as shown in the following example.

7 Click to select the following configuration information:

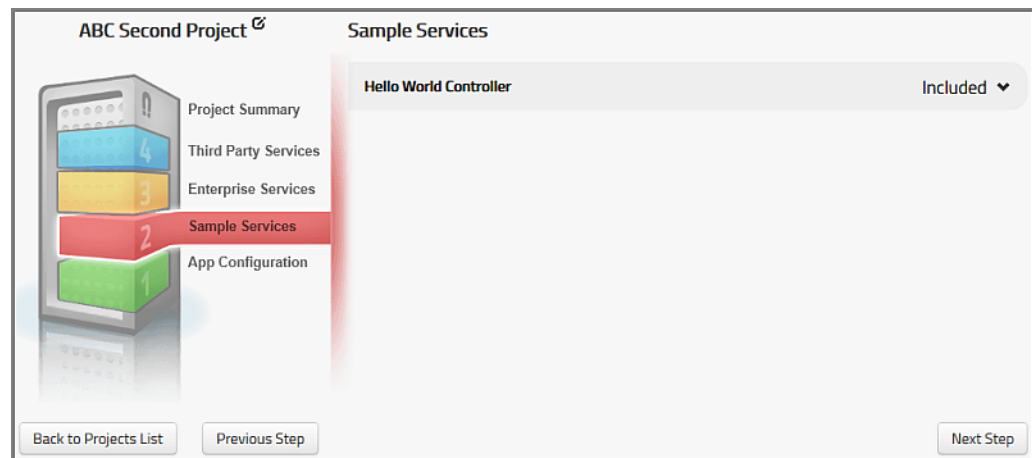
- *Encryption Services*. This is used to encrypt security sensitive information in your configuration as well as on your mobile devices.
- *Location data collection*: This enables geo-tagged coordinates to be transported each time a controller is invoked on your server, allowing your server to know the location of the mobile app user.



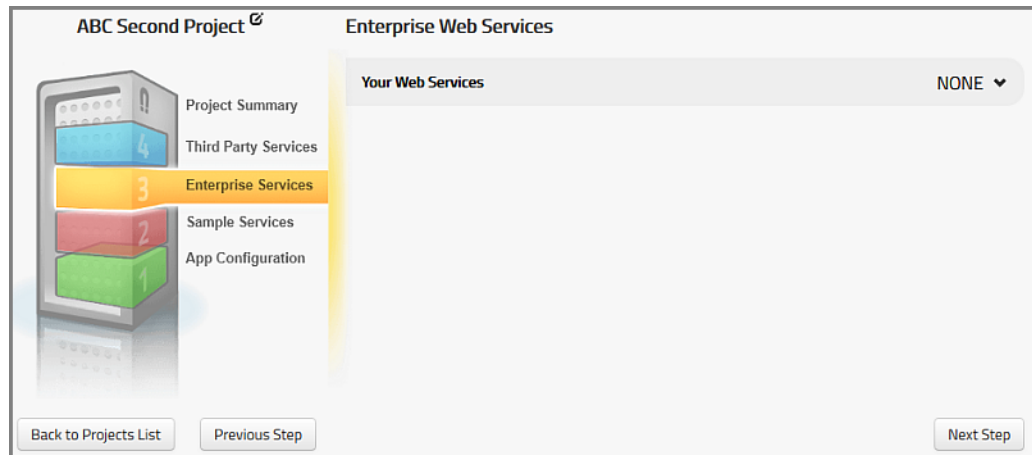
- *User Authentication*: You can select the appropriate data store from where you want to read user credentials to authenticate for accessing server from the mobile clients.
 - Select Default User if you want the Developer Factory to provide a user name and password for you that will get you access to a deployed sandbox.
 - Select the appropriate option if you want to use your own tools and environment to control access to the sandbox. User name and password will be managed by one of these three systems.
- 8 Click **Include** to enable these services so you can utilize those notification capabilities for your apps.
 - Google Android Notification Service
 - Apple iOS Notification Service
 - Email Notification Service
 - 9 Click **Next Step** to select the sample services.
 - 10 Include the Hello World Controller if you want to get a template to start developing your own custom controllers. However, if you do not want to add your own custom controllers, then click **Don't Include**.



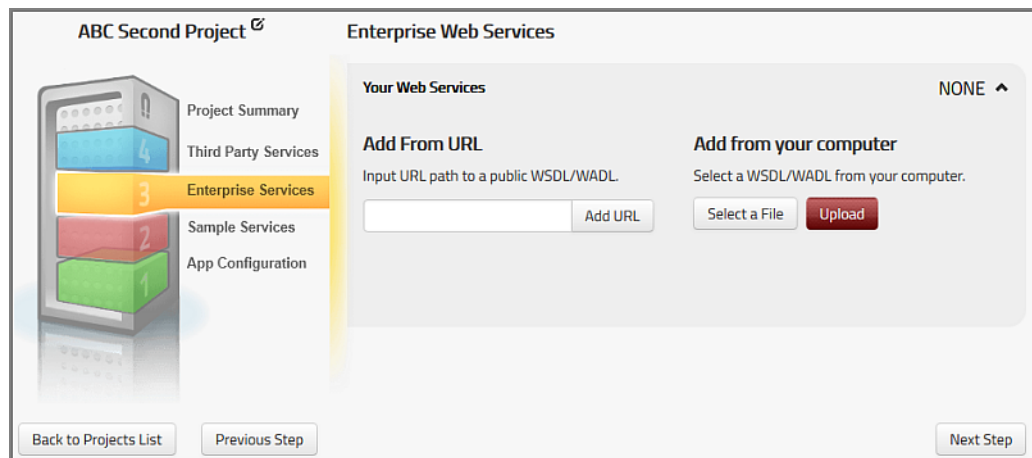
The Hello World Controller provide a simple UI that can be run on an emulator or on a phone and connect to the deployed sandbox (or any server where you may have installed our generated server JAR).



11 Click **Next Step** to add your Web services.



12 Click **NONE** to display the Web Services options.



13 Enter the URL to add your WSDL or WADL file, click **Add URL**.

— Or —

Click **Select a File** to add your WSDL or WADL file from your computer, click **Upload**.

14 Configure security settings for your app to properly communicate with this Web service.



The security configuration window does not appear if security configuration is included in your WSDL file.



The screenshot shows the 'Enterprise Web Services' configuration interface. On the left, a sidebar lists four categories: 'Third Party Services' (4), 'Enterprise Services' (3), 'Sample Services' (2), and 'App Configuration' (1). The 'Enterprise Services' category is selected. The main area is titled 'Your Web Services' and shows '1 Services'. It has two tabs: 'Add From URL' and 'Add from your computer'. The 'Add From URL' tab is active, showing a text input for 'Input URL path to a public WSDL/WADL.' and an 'Add URL' button. Below this, a 'WSDL' section shows the URL 'http://raspi7.magnet.c...' and a warning 'No WS Policy detected. Select a setting below:'. There are four radio button options: 'Basic Auth' (selected), 'UsernameToken', 'None', and 'Specify later'. Each option has a description. At the bottom, there are buttons for 'Back to Projects List', 'Previous Step', and 'Next Step'.

- When Basic Auth is selected, the following window appears:

This screenshot shows the configuration window for 'Basic Auth'. It has the same WSDL URL and warning as the previous window. The 'Basic Auth' radio button is selected. Below the radio buttons, there are two text input fields for 'ID' and 'Password'. There are two buttons for 'Timestamp ON' and 'Timestamp OFF', with 'Timestamp OFF' being highlighted. A note at the bottom says 'You may leave Username and Password blank if you want to specify them later.'

- When UsernameToken is selected, the following window appears:

This screenshot shows the configuration window for 'UsernameToken'. It has the same WSDL URL and warning. The 'UsernameToken' radio button is selected. Below the radio buttons, there are two text input fields for 'ID' and 'Password'. There are two buttons for 'cleartext' and 'digest', with 'digest' being highlighted. There are also two buttons for 'Timestamp ON' and 'Timestamp OFF', with 'Timestamp OFF' being highlighted. A note at the bottom says 'You may leave ID and Password blank if you want to specify them later.'



15 Click **Next Step** to display the *3rd Party Services* page.

16 Select the third-party services that you want to include in your application, click **Include**.

17 Optionally enter the credentials and password (you received those from Connected Apps) in each of the services you include.

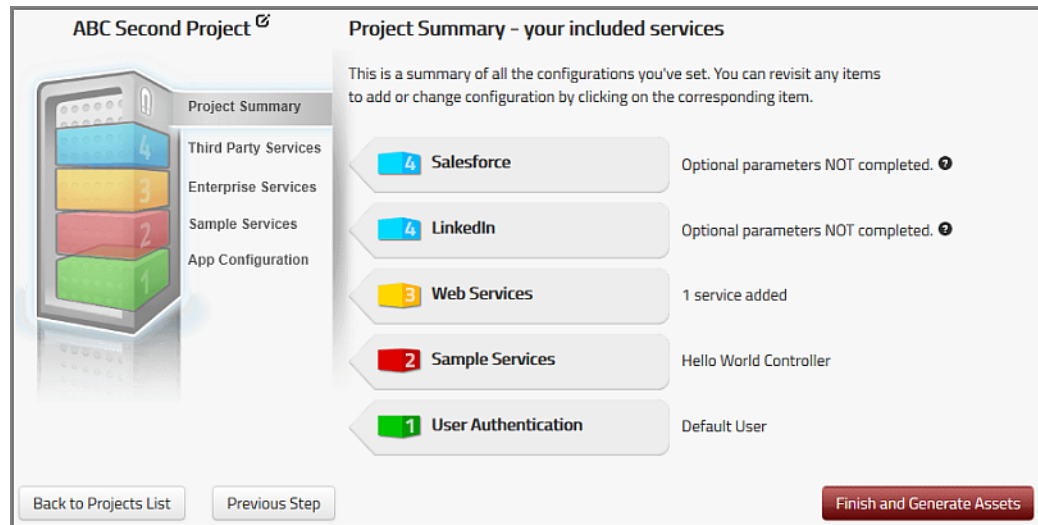


You can configure these settings later in your generated project.



18 Click **Next Step**.

Your project summary is displayed as shown in the following example.

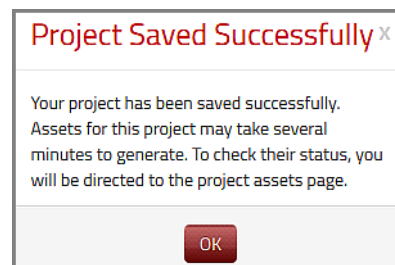


The color-coded buttons shown on the Project Summary side are now interactive. Clicking them takes you directly to the intended page. For example, clicking User Authentication opens the App Configuration step.

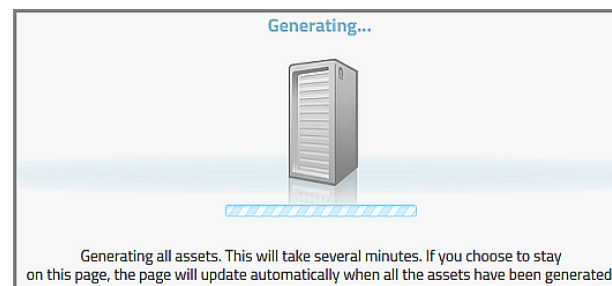
19 Review your project.

- To make changes, click **Previous Step** to the step that you need changing.
- Click **Finish and Generate Assets** if no further changes are needed.

A dialog box appears informing of the project status.



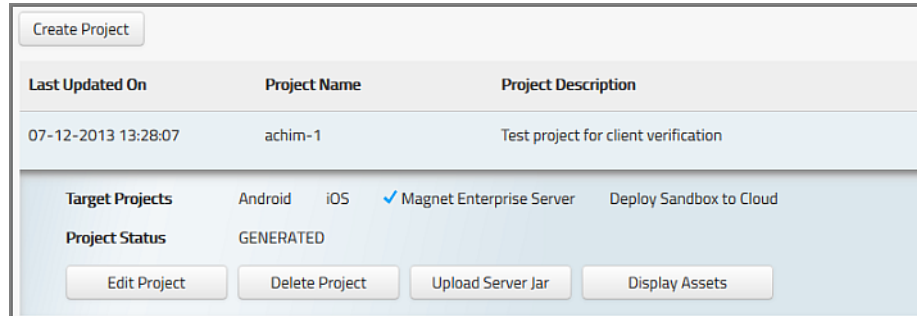
20 Click **OK** to start the asset generation process.



Displaying the Generated Assets

After your project is generated, you can download it to your local machine for customization or deploy it to the sandbox for testing.

- 1 Click the **Projects** tab, and select the intended project.



Last Updated On	Project Name	Project Description
07-12-2013 13:28:07	achim-1	Test project for client verification

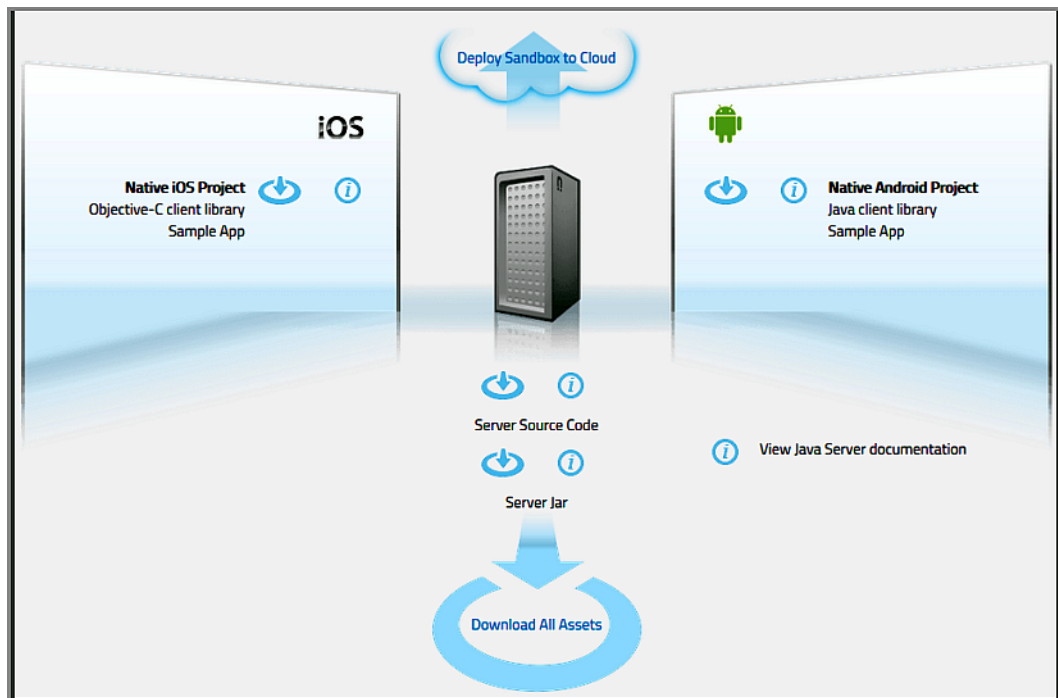
Target Projects	Android	iOS	<input checked="" type="checkbox"/> Magnet Enterprise Server	Deploy Sandbox to Cloud
Project Status	GENERATED			

[Edit Project](#)
[Delete Project](#)
[Upload Server Jar](#)
[Display Assets](#)

- 2 Click **Display Assets**.



You are informed if the project has not been generated. Click **Generate Assets**, then repeat step 2.



- 3 Click **Deploy Sandbox to Cloud**, or select the project you want to download, or click **Download All Assets** to download all assets.

Each server project is compressed into a zip file.



Modifying a Project



Do not change the maven groupId, artifactId, and version of your project if you plan to regenerate the mobile assets out of a modified project.

- 1 Sign in to your Developer Factory account.

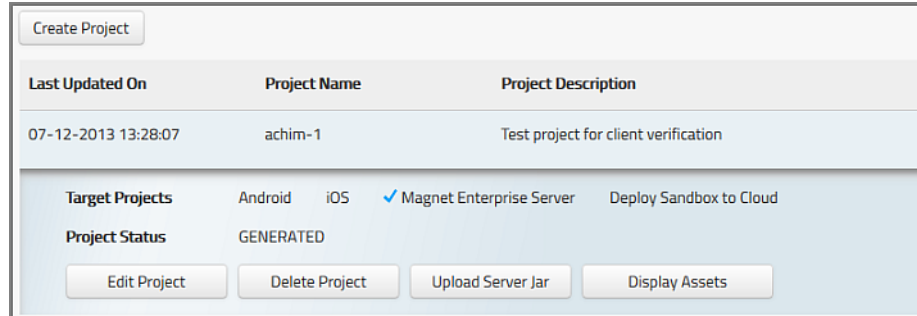


- 2 Click the **Projects** tab to display all projects that you have created, as shown in the following example.

Create Project		
Last Updated On	Project Name	Project Description
07-05-2013 11:02:41	ABC Project	WSDL
07-03-2013 17:21:12	hoa-july-4th-project	Test QA dev factory project
07-03-2013 14:25:31	jim 3	test
07-03-2013 13:43:47	hoa-weblogic-awesomeservice	project to celebrate July 4th



3 Select the project you want to modify.



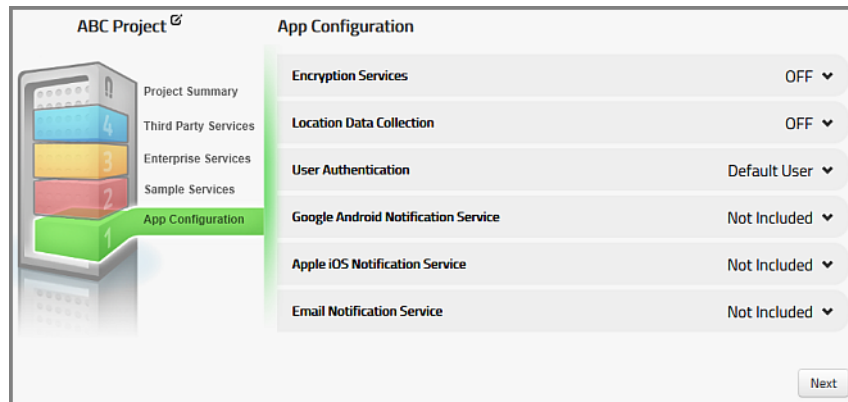
The screenshot shows a web interface for managing projects. At the top is a 'Create Project' button. Below it is a table with three columns: 'Last Updated On', 'Project Name', and 'Project Description'. The table contains one row with the values '07-12-2013 13:28:07', 'achim-1', and 'Test project for client verification'. Below the table, there are sections for 'Target Projects' (listing Android, iOS, and Magnet Enterprise Server with a checkmark) and 'Project Status' (showing GENERATED). At the bottom are four buttons: 'Edit Project', 'Delete Project', 'Upload Server Jar', and 'Display Assets'.

4 Click **Edit Project**.

A page similar to the following example appears.



All color-coded buttons are now interactive. Clicking them takes you directly to the intended page.



The screenshot shows the 'ABC Project' App Configuration page. On the left is a sidebar with a stack of four colored boxes labeled 1, 2, 3, and 4, corresponding to 'App Configuration', 'Sample Services', 'Enterprise Services', and 'Third Party Services' respectively. The main area is titled 'App Configuration' and contains several settings: 'Encryption Services' (OFF), 'Location Data Collection' (OFF), 'User Authentication' (Default User), 'Google Android Notification Service' (Not Included), 'Apple iOS Notification Service' (Not Included), and 'Email Notification Service' (Not Included). A 'Next' button is at the bottom right.

5 Click the intended color-coded services layers and modify any information (refer to *Creating a Project* for detailed description).



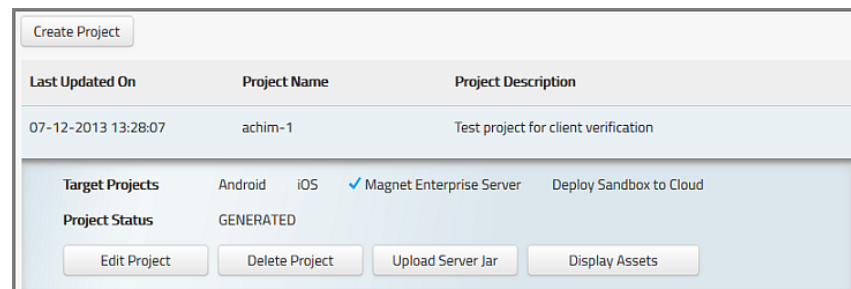
Uploading a Project

When your project customization is complete, you need to upload the project from your local machine to the Developer Factory, and regenerate the assets.

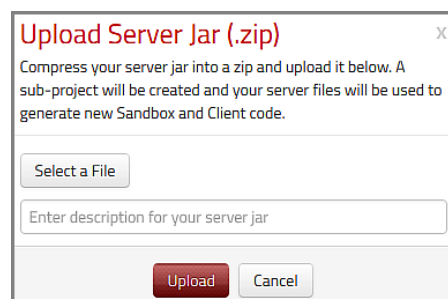
- 1 Sign in to your Developer Factory account.



- 2 Click the **Projects** tab to display all projects that you have created.
- 3 Select the project you want to upload.



- 4 Click **Upload Server Jar**.



- 5 Click **Select a File**, and navigate to the *aws-Server/target* directory where the server JAR zip is stored.



Refer to *Chapter 9, Compiling Your Project*, for the directory structure of the project.

- 6 Enter an optional description for the server JAR.
- 7 Click **Upload**.

Removing a Project

When you close your Developer Factory account, all projects that you owned will be automatically removed, and you will be removed from all projects to which you had been invited.

- 1 Sign in to your Developer Factory account.

The screenshot shows the Developer Factory web interface. At the top is a navigation bar with links: Developer Factory, Projects, Resources, Documentation, Support, and a user profile icon with the email doris.lok@magnet.com. Below the navigation bar is a large section titled "Introducing Magnet Developer Factory". The text below the title says: "Get your projects started in the Developer Factory by quickly generating a custom SDK for your mobile apps. First, configure the services you want to enable for your mobile apps, followed by specifying the web services you want to use. That's it! The Developer Factory automatically constructs your own Magnet Mobile App Server along with the matching SDKs for iOS and Android app development." Below this text is a section titled "How it works:" followed by a five-step process flow diagram. The steps are: 1. Configure (Set up a new project using our project wizard), 2. Sample (Choose sample services to give yourself a head start), 3. Select (Connect your project to SOAP or REST web services), 4. Include (Add 3rd party services to your project), and 5. Deploy (Your Magnet sandbox gets deployed). At the bottom of the process flow is a button labeled "Start a new project".



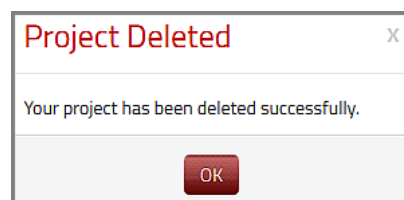
- Click the **Projects** tab to display all projects that you have created, as shown in the following example.

Create Project		
Last Updated On	Project Name	Project Description
07-05-2013 11:02:41	ABC Project	WSDL
07-03-2013 17:21:12	hoa-july-4th-project	Test QA dev factory project
07-03-2013 14:25:31	jim 3	test
07-03-2013 13:43:47	hoa-weblogic-awesomeservice	project to celebrate July 4th

- Select the project you want to remove.

Create Project		
Last Updated On	Project Name	Project Description
07-12-2013 13:28:07	achim-1	Test project for client verification
<div>Target Projects</div> <div>Android iOS <input checked="" type="checkbox"/> Magnet Enterprise Server Deploy Sandbox to Cloud</div> <div>Project Status</div> <div>GENERATED</div> <div>Edit Project Delete Project Upload Server Jar Display Assets</div>		

- Click **Delete Project**.
A dialog box appears asking for confirmation.
- Click **Continue**.
A dialog box appears showing the operation status.



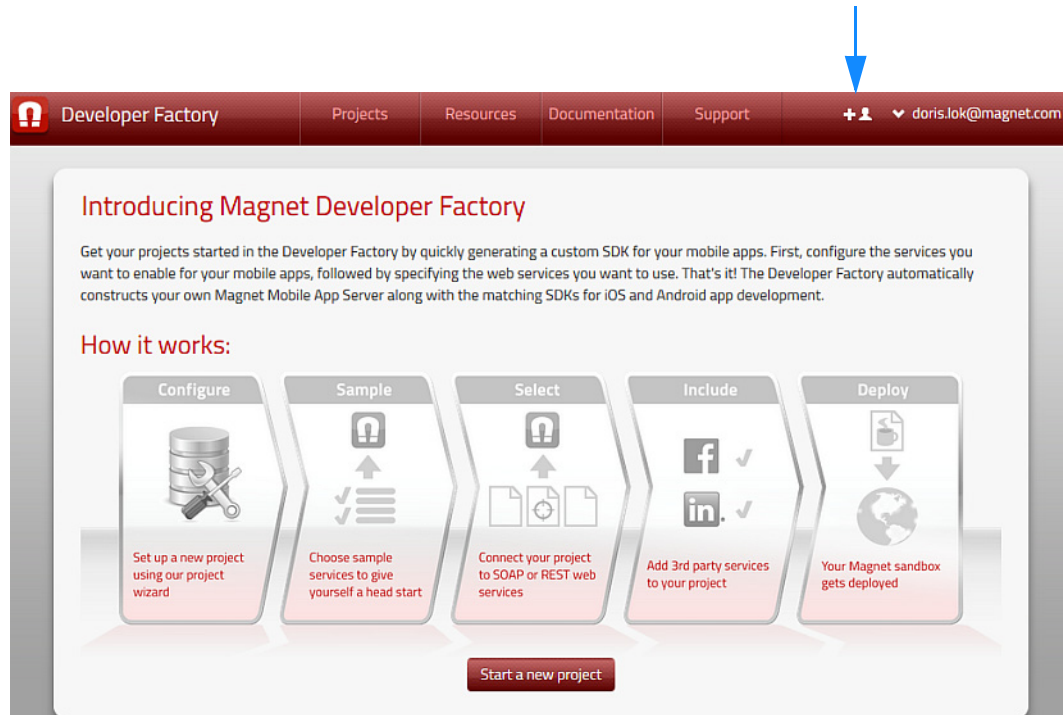
- Click **OK**.
The project is immediately removed from the project page.



Inviting Other Developers

You can invite other developers to the Magnet Developer Factory to create their own accounts. If your invitees are not registered with Developer Factory, they need to first register for an account.

- 1 Sign in to your Developer Factory account.



- 2 Click **+ person** to display the invitation, as shown in the following:

The screenshot shows a dialog box titled 'Invite Others' with a close button (X) in the top right corner. The dialog contains the following elements:

- A text prompt: 'Enter the email address of the individual you would like to invite to the Magnet Developer Factory.'
- An input field labeled 'Enter Email Address'.
- A section titled 'Message to include' with a text area containing the message: 'I found this great new tool to quickly create mobile apps for me.'
- At the bottom, there are two buttons: 'Send Invitation' and 'Close'.



3 Enter the e-mail address of the person you want to invite, and optionally delete the existing message from the **Message to include** box, and enter your own message that you want to include with your e-mail.

4 Click **Send Invitation**.

You are informed of the operation status. Your recipient will receive an e-mail notification that includes a link to register an account.



MMLoginService is an interface that provides methods for user log in and log out operations. The following table lists and describes each method and the page reference pointing to the method.

Methods	Description	Page
loginWithCredentials	This method requires users to provide credentials when logging in.	page 28
loginWithSavedCredentials	This method authenticates users with saved credentials.	page 29
logout	This method logs out the user.	page 29

loginWithCredentials

The method is an asynchronous operation with callback blocks (that is, the calling thread does not block for the call to be executed). Use this method when users are required to provide credentials when logging in.

```
- (void)loginWithCredentials:(NSDictionary *)credentials  
    block:(void (^)(BOOL success))block;
```

Parameter

- *credentials* indicates a dictionary containing credentials.
- *block* indicates a block object that is to be executed when the request succeeds. This block has no return value and takes one argument, `BOOL`.

Return

Authentication success or failure.

Example

```
NSString* hostName = "myhostname";  
NSString* protocol = "http";  
NSInteger port = 80;  
NSString* alias = @"rest";  
NSString* serverAddress = [protocol stringByAppendingFormat:@"%://%@:%i/%@",  
    hostName, port, alias];  
  
NSDictionary* dict = @{@"username":@"myusername", @"password":@"mysecret",  
    @"serverAddress":serverAddress, @"useSSL":@"NO",  
    @"trustAllCertificates":@"NO"};  
  
[[MMLoginService instance] loginWithCredentials:dict block:^(BOOL success) {  
    if (success) {  
        // my success callback  
    } else {  
        // my failure callback  
    }  
}  
};
```



loginWithSavedCredentials

The method is an asynchronous operation with callback blocks (that is, the calling thread does not block for the call to be executed). Use this method when users are authenticated with saved credentials when logging in.

```
- (void)loginWithSavedCredentials:(void (^)(BOOL success))block;
```

Parameter

- *block* indicates a block object that is to be executed when the request succeeds. This block has no return value and takes one argument, BOOL.

Return

Authentication success or failure.

Example

```
[[MMLoginService instance] loginWithSavedCredentials:^(BOOL success)
{
    if (success) {
        // my success callback
    } else {
        // my failure callback
    }
}];
```

Logout

This method logs out the current user.

```
- (void)logout;
```

Parameter

None.

Return Value

Not applicable.

Example

```
[[MMLoginService instance] logout];
```





6. Defining Caching and Off-line Mode Services

The AsyncMyController represents functionality that you write running on your server instance(s). The AsyncMyController allows you to invoke a request in an off-line mode and caching services remotely from your mobile client.

A request may be reliable or asynchronous. A reliable request is associated with the name of a queue, and queues are processed sequentially. An asynchronous request indicates that no queue is allowed.

Constraints are used to control the queue process, and may be applied to both reliable and asynchronous requests. In the case with a reliable request, only when the constraint associated with the head of the queue is in an allowed state can the queue be drained of its elements—meaning that the invocations are being messaged to the server asynchronously and reliably.

When an asynchronous request has a constraint and the constraint is dis-allowed, processing will continue to the next element of the queue. The element is not allowed because the constraint is essentially skipped and will be retried later, provided the request timeout on the entry has not expired the next time around.

The following example shows code used in the AsyncMyController.

```
@interface AsyncMyController : MMOpusController

+ (MMCall *)localizedCategoriesWithOptions:(MMOptions *)options
    success:(void (^)(NSArray *categories))success
    failure:(void (^)(NSError *error))failure;

+ (MMCall *)updateCategoryWithOptions:(MMOptions *)options
    success:(void (^)(void))success
    failure:(void (^)(NSError *error))failure;

@end
```

Declaring Options

The following two classes are used to determine how the reliable and asynchronous calls are handled.

- MMReliableCallOptions
- MMAsyncCallOptions

Creating MMReliableCallOptions

When the MMReliableCallOptions class is created, you can include methods that allow the caller to:

- Use the cached value to queue up a call in a persistent storage.
- Execute the calls in sequential manner.
- Impose a restriction when the call can be invoked.
- Specify a timeout for this call.
- Specify a static callback.

The following example shows MMReliableCallOptions is created with two methods. Both cacheAge and requestTimeout are set at 3600 seconds, and cached value is used in an off-line mode. The call returns success or failure with userInfo that includes callId, someUUID, and token (if defined).

```
MMReliableCallOptions *options = [MMReliableCallOptions new];
NSTimeInterval cacheAge = 3600;
[options setCacheAge:cacheAge ignoreIfOffline:NO];
NSTimeInterval requestTimeout = 1 * 60 * 60;
[options setRequestTimeout:requestTimeout];
[AsyncMyController updateCategoryWithOptions:options success:nil
failure:nil];
```



Setting Cache Age

The *setCacheAge* method specifies an acceptable cache age and enables caching the result. The call uses the cached value (if available) when its age is within the timeout interval; otherwise, the cached value is discarded and the new result is cached.

If this option is not specified, the cached value is discarded and the new result is not cached. You can specify in the *ignoreIfOffline* parameter whether to allow the caller to use the cached value (if available) in an off-line mode.

Parameter

- *cacheAge* indicates the length of time in seconds that cache remains valid.
- *ignoreIfOffline* indicates whether or not to use the cached value in an off-line mode.
 - NO indicates that the cached value is used in an off-line mode.
 - YES indicates otherwise.

Setting Request Timeout

The *setRequestTimeout* method specifies a timeout value when the call object expires. You can specify a longer timeout value to survive an off-line mode

Parameter

- *requestTimeout* indicates the request timeout expressed in seconds.

Creating MMAsyncCallOptions

This class is used for an asynchronous call, which allows the caller to use the cached value, impose a restriction when the call can be invoked, and specify a callback. The caller must use **create()** to create an instance of the options.

The following example shows MMAsyncCallOptions is created with two methods. The *cacheAge* is set at 3600 seconds, and cached value is not used in an off-line mode. A restriction is imposed with the *MMWifiConstraint* class. The call returns success or failure.



```
MMAsyncCallOptions *options = [MMAsyncCallOptions new];
NSTimeInterval cacheAge = 1 * 60 * 60;
[options setCacheAge:cacheAge ignoreIfOffline:YES];
[options setConstraint:[MMWifiConstraint class]];
[AsyncMyController localizedCategoriesWithOptions:options
success:^(NSArray *categories){
    NSLog(@"categories = %@", categories);
} failure:^(NSError *error) {
    NSLog(@"Oops! Failed to get categories!");
    NSLog(@"error = %@", error);
}];
```

Setting Cache Age

The *setCacheAge* method specifies an acceptable cache age and enables caching the result. The call uses the cached value (if available) when its age is within the timeout interval; otherwise, the cached value is discarded and the new result is cached.

If this option is not specified, the cached value is discarded and the new result is not cached. You can specify in the *ignoreIfOffline* parameter whether to allow the caller to use the cached value (if available) in an off-line mode.

Parameter

- *cacheAge* indicates the length of time in seconds that cache remains valid.
- *ignoreIfOffline* indicates whether or not to use the cached value in an off-line mode.
 - NO indicates that the cached value is used in an off-line mode.
 - YES indicates otherwise.

Setting Constraints

Constraints provide a means to impose conditions that a queue must meet before it can be processed. The *setConstraint* method specifies a constraint to invoke a queued call. A queued call can be processed only when the constraint is met. The current release supports one restriction, *MMWifiConstraint*.



Making a Reliable Request

It is typically the case that invocations from the mobile client occur asynchronously and reliably. They are asynchronous in that the invocation is recorded for later playback to the server on a separate worker thread. They are reliable in that after an invocation has been submitted, it is guaranteed to be run once—and only once—on the server side.

You can make a reliable request with the following specifications:

- Cache results and callback (page 35)
- Cache results but no callback (page 35)
- No caching and no callback (page 35)

Caching Results and Callback

```
MMReliableCallOptions *options = [MMReliableCallOptions new];
NSTimeInterval requestTimeout = 1 * 60 * 60
[options setRequestTimeout:requestTimeout];
NSTimeInterval cacheAge = 1 * 60 * 60;
[options setCacheAge:cacheAge ignoreIfOffline:NO];
[AsyncMyController updateCategoryWithOptions:options success:nil
failure:nil];
```

Caching Results but no Callback

```
MMReliableCallOptions *options = [MMReliableCallOptions new];
NSTimeInterval requestTimeout = 10 * 60 // in seconds
[options setRequestTimeout:requestTimeout];
[options setCacheAge:cacheAge ignoreIfOffline:NO];
[AsyncMyController updateCategoryWithOptions:options success:nil
failure:nil];
```

No Caching and No Callback

```
MMReliableCallOptions *options = [MMReliableCallOptions new];
NSTimeInterval requestTimeout = 10 * 60 // in seconds
[options setRequestTimeout:requestTimeout];
[AsyncMyController updateCategoryWithOptions:options success:nil
failure:nil];
```



Making an Asynchronous Request

An asynchronous request indicates that no queue is named explicitly; consequently, a queue is not allowed. There might be occasions to make a request without caching the results or reliability is not a concern. You can make an asynchronous request with the following specifications:

- Caching results and callback but no reliability (page 36)
- No caching, reliability, or callback (page 36)
- No caching or reliability but with callback (page 36)

Caching Results and Callback but no Reliability

```
MMAsyncCallOptions *options = [MMAsyncCallOptions new];
NSTimeInterval cacheAge = 1 * 60 * 60; // in seconds
[options setCacheAge:cacheAge ignoreIfOffline:YES];
[AsyncMyController localizedCategoriesWithOptions:options
success:^(NSArray *categories){
    NSLog(@"categories = %@", categories);
} failure:^(NSError *error) {
    NSLog(@"Oops! Failed to get categories!");
    NSLog(@"error = %@", error);
}];
```

No Caching, Reliability, or Callback

```
[AsyncMyController updateCategoryWithOptions:nil success:nil
failure:nil];
```

No Caching or Reliability but With Callback

```
[AsyncMyController localizedCategoriesWithOptions:nil success:^(NSArray
*categories){
    NSLog(@"categories = %@", categories);
} failure:^(NSError *error) {
    NSLog(@"Oops! Failed to get categories!");
    NSLog(@"error = %@", error);
}];
```



Applying a Constraint to an Asynchronous Request

Constraints are useful so that processing of the queue can be paused under varying network conditions. For instance, some operations should occur only in a WIFI zone. The call fails fast and the response listener is immediately called when the constraint is not met.

The following example shows a constraint is applied to an asynchronous request with caching results and callback.

```
MMAsyncCallOptions *options = [MMAsyncCallOptions new];
NSTimeInterval cacheAge = 1 * 60 * 60; // in seconds
[options setCacheAge:cacheAge ignoreIfOffline:YES];
[options setConstraint:[MMWifiConstraint class]];
[AsyncMyController localizedCategoriesWithOptions:options
success:^(NSArray *categories){
    NSLog(@"categories = %@", categories);
} failure:^(NSError *error) {
    NSLog(@"Oops! Failed to get categories!");
    NSLog(@"error = %@", error);
}];
```





The Magnet App Server currently provides three social controllers that you can include in your app to enable access to third-party services. These controllers include Salesforce, LinkedIn, and Facebook.

Using the Salesforce Controller

The following code example makes the search call, which returns contacts, leads, and accounts. The search string specifies the text expression to search for, the scope of fields to search, the list of objects, and fields to retrieve.

```
SearchNode *searchNode = [SearchNode new];
searchNode.searchString = @"find {4159017000} in phone fields returning
contact(id, phone, firstname, lastname), lead(id, phone, firstname, lastname),
account(id, phone, name)";
SoapController *soapController = [SoapController new];
[soapController search:searchNode success:^(SearchResponseNode *response) {
    NSLog(@"response = %@", response);
    SearchRecordNode *searchRecordNode = (SearchRecordNode *)
response.result.searchRecords[0];
    AccountNode *account = (AccountNode *) searchRecordNode.record;
    NSLog(@"name: %@, phone: %@", account.name.element, account.phone.element);
} failure:^(NSError *error) {
    NSLog(@"error = %@", [error localizedDescription]);
}];
```

Using a Facebook Controller

The Facebook controller enables searching for friends and sharing updates from a mobile device.

```
[[FacebookController new] friendsIterator:^(MMListResponse *response) {
    NSLog(@"response = %@", response);
    __block NSMutableArray *friendNames = [NSMutableArray
arrayWithCapacity:[response.page count]];
    [response.page enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL
*stop) {
        User *user = (User *) obj;
        [friendNames addObject:user.name];
    }];
    UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:@"Facebook friends!"
message:[friendNames componentsJoinedByString:@" ", ""]
delegate:nil
cancelButtonTitle:@"OK"
otherButtonTitles:nil];
    [alert show];
}
failure:^(NSError *error) {
    NSLog(@"error = %@", [error localizedDescription]);
    UIAlertView *alert = [[UIAlertView alloc]
initWithTitle:@"Oops!"
message:@"Cannot get Facebook friends! Authenticate with Facebook."
delegate:nil
cancelButtonTitle:@"OK"
otherButtonTitles:nil];
    [alert show];
}];
```



The iOS apps are built using the Model-View-Controller (MVC) design pattern that assigns objects in an application one of three roles: model, view, or controller. This chapter provides information about the controller object.

Sending a Message to a Controller

Most controllers use the same construct. Each LoginController method is an asynchronous call with callback blocks (that is, the calling thread does not block for the call to be executed); therefore, the LoginController has a slightly different construct from the corresponding server implementation. Refer to *Chapter 5, Defining Log In and Out Services*, for information.

Use the following construct to send a message to a controller:

```
<controller> *controllerInstance = [<controller> new];
```

The following example shows YellowPagesPortTypeController being added:

```
YellowPagesPortTypeController *controller = [YellowPagesPortTypeController new];
```

Each application controller is represented by an interface similar to that provided by the server as shown in the following modified construct:

```
- (void) methodName:<list of params>
    success: (void (^)(<optional return type>)) success
    failure: (void (^)(NSError *)) failure;
```

The following example shows the YellowPagesPortTypeController interface definition:

```
- (void)findAll:(FindRequestNode * )findAll
    success:(void (^)(ListOfFindResponseNode *response))success
    failure:(void (^)(NSError *error))failure;

- (void)match:(NSString * )match
    success:(void (^)(ListOfFindResponseNode *response))success
    failure:(void (^)(NSError *error))failure;

- (void)find:(FindRequestNode * )find
    success:(void (^)(FindResponseNode *response))success
    failure:(void (^)(NSError *error))failure;
```

Using Node

A corresponding Objective-C object is generated by the Developer Factory for the node that is exposed by the server. If the server exposes a FindResponseNode, a FindResponseNode object will be available in Objective-C. The Magnet Mobile Server API on iOS is capable of marshaling and un-marshaling such node.

The following example shows an interface definition for FindResponseNode:

```
- (void)findAll:(FindRequestNode * )findAll
    success:(void (^)(ListOfFindResponseNode *response))success
    failure:(void (^)(NSError *error))failure;

- (void)match:(NSString * )match
    success:(void (^)(ListOfFindResponseNode *response))success
    failure:(void (^)(NSError *error))failure;

- (void)find:(FindRequestNode * )find
    success:(void (^)(FindResponseNode *response))success
    failure:(void (^)(NSError *error))failure;
```



Using MMDData

The MMDData object represents data. You use the @property interface to declare the data type associated with the MMDData object.

The following example shows an interface definition for MMDData:

```
@property (nonatomic, copy) NSString *mimeType;  
  
@property (nonatomic, strong) NSData *binaryData;  
  
@property (nonatomic, copy) NSString *uri;
```





Your iOS app is delivered as an Objective-C language project, Xcode-ready. A zip file that includes the Xcode project, mobile server framework, client stubs for the controller, and a sample jump start app.

The ClientLibrary items are automatically generated and the following directories are created:

- Entities: contains entities in the Magnet Mobile App Server.
- EntityResources: contains helper classes (one per entity) for interacting with entities used for CRUD (Create, Read, Update, Delete) operations.
- REST/Client: contains a wrapper class for RestKit called RestClient. (You should not need to interact with this class directly.)
- REST/Response: contains classes that represent the JSON output from the Magnet Mobile App Server.

After you have downloaded your project from the Developer Factory to your local environment, you can write your app based on the sample jump start app, then upload the new app to the Developer Factory for new assets generation.



A timely response to issues from customers can contribute to the success of an IT Support organization. TechCare app enables an administrator to view tickets as they come in, and to immediately take appropriate actions.

This chapter shows use cases and sample code used in the TechCare app.

Fetching Users

The following example shows that users are fetched and sorted in an ascending order.

```
[[Administration new] adminUsers:@"false" options:nil success:^(NSMutableArray *response) {  
}  
  
failure:^(NSError *error) {  
}];
```

Caching Information

The following example shows to get and cache the current user information.

```
[[Administration new] adminCurrentUserWithOptions:nil success:^(User *response)  
{  
  
} failure:^(NSError *error) {  
}];
```

Posting Comments

The following example shows to post comments using `MMReliableCallOptions`. Although comments can be posted offline, this example shows a reliable request is made, which means the comment can be posted only when there is connectivity. It also shows that the request will expire after 7 days.

```
MMReliableCallOptions *options = [MMReliableCallOptions new];
options.requestTimeout = 86400 * 7;

[[Administration new] adminAddComment:self.ticket.magnetId status:comment
options:options success:^(Comment *response) {

} failure:^(NSError *error) {

}];
```

Updating Tickets

The following example shows to update tickets using `MMReliableCallOptions`. Although tickets can be updated offline, this example shows a reliable request is made, which means the tickets can be updated only when there is connectivity.

```
MMReliableCallOptions *options = [MMReliableCallOptions new];

[[Administration new] adminStartWorkOnTicketTicket:self.ticket.magnetId
options:nil success:^(Ticket *response) {

} failure:^(NSError *error) {

}];
```



Receiving Notifications (via APNS)

The following example shows to register APNS token for push notifications.

```
[[Administration new] adminRegisterDevice:@"apns"
    deviceToken:deviceToken
    options:nil
    success:^(NSString *response) {
    }
    failure:^(NSError *error) {
    }
];
```

Setting a Geo Fence for Android Apps

The following example shows to create geo fencing. (This fence is used as a constraint for the Android app, it does not apply to the TechCare app).

```
[[Administration new] adminCreateGeofence:@latitude
    longitude:@longitude
    radius:@radius
    isInteriorBounds:@"true"
    options:nil
    success:^(Geofence *response) {
    }
    failure:^(NSError *error) {
    }
];
```



