Kony Fabric

Sync Services Design Guidelines

Release V8

Document Relevance and Accuracy

This document is considered relevant to the release stated on this title page and the document version stated on the Revision History page. Remember to always view and download the latest document version relevant to the software release you are using.

© 2013 by Kony, Inc. All rights reserved

Copyright © 2013 by Kony, Inc.

All rights reserved.

September, 2017

This document contains information proprietary to Kony, Inc., is bound by the Kony license agreements, and may not be used except in the context of understanding the use and methods of Kony, Inc., software without prior, express, written permission. Kony, Empowering Everywhere, Kony Fabric, Kony Nitro, and Kony Visualizer are trademarks of Kony, Inc. MobileFabric is a registered trademark of Kony, Inc. Microsoft, the Microsoft logo, Internet Explorer, Windows, and Windows Vista are registered trademarks of Microsoft Corporation. Apple, the Apple logo, iTunes, iPhone, iPad, OS X, Objective-C, Safari, Apple Pay, Apple Watch, and Xcode are trademarks or registered trademarks of Apple, Inc. Google, the Google logo, Android, and the Android logo are registered trademarks of Google, Inc. Chrome is a trademark of Google, Inc. BlackBerry, PlayBook, Research in Motion, and RIM are registered trademarks of BlackBerry. SAP® and SAP® Business Suite® are registered trademarks of SAP SE in Germany and in several other countries. All other terms, trademarks, or service marks mentioned in this document have been capitalized and are to be considered the property of their respective owners.

Revision History

Date	Document Version	Description of Modifications/Release
09/08/2017	1.0	Document updated for release V8. Worked on rebranding of MobileFabric to Kony Fabric.

© 2013 by Kony, Inc. All rights reserved

Table of Contents

© 2013 by Kony, Inc. All rights reserved

© 2013 by Kony, Inc. All rights reserved

1. Preface

Kony Fabric Sync Framework consists of Kony Fabric Sync Client and Kony Fabric Sync Server. Kony Fabric Sync Management Console is an aid to connect to the client and communicate with Kony Fabric Sync Server.

Kony Fabric Sync Management Console provides a single point of control for monitoring and configuring the Kony Fabric Sync Console. It includes data integration, storage, and analysis resulting in reduced training time, manual work, and upholding standard operating procedures.

Kony Fabric Sync Services are the services that synchronize data between client and enterprise data source bi-directionally.

1.1 Purpose

This document provides detailed guidelines required to design Kony Fabric Sync good user experience during initial sync, SyncScopes and SyncObjects, to identify the appropriate SyncStrategy, and guidelines to be followed while implementing web services, databases and SAP DataSource connector to integrate with Kony Fabric Sync Server.

1.2 Intended Audience

This document is intended for developers or system administrators who are responsible for installing and deploying Kony Fabric Sync.

1.3 Formatting Conventions

The following typographical conventions are used throughout the document:

© 2013 by Kony, Inc. All rights reserved

Convention	Explanation
`Monospace`	■ User input text, system prompts, and responses ■ File path ■ Commands ■ Program code ■ File Names.
Italic	■ Emphasis ■ Names of books, and documents ■ New terminology.
Bold	■ Windows ■ Menus ■ Buttons ■ Icons ■ Fields ■ Tabs.
URL	Active link to a URL
Note	Provides helpful hints or additional information
Important	Highlights actions or information that might cause problems to systems or data

1.4 Contact Us

We welcome your feedback on our documentation. Write to us at techpubs@kony.com.

© 2013 by Kony, Inc. All rights reserved

For technical questions, suggestions, comments or to report problems on Kony product line, contact support@kony.com.

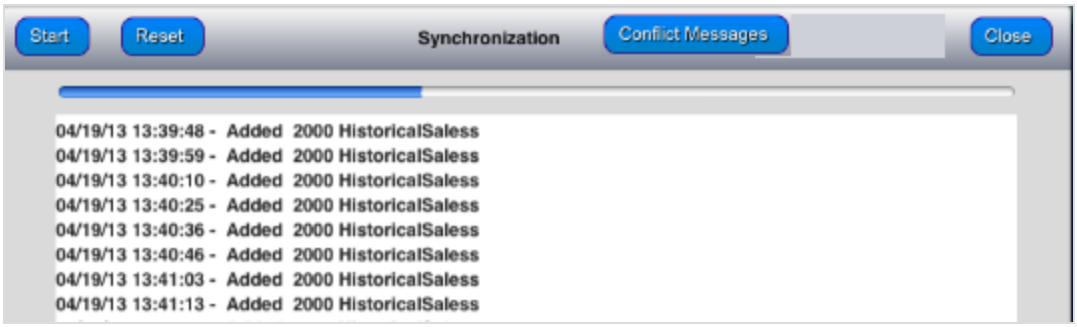
© 2013 by Kony, Inc. All rights reserved

2. Designing a Good User Experience

You need to design a good user experience during the sync process, specially for the initial sync.

Applications that are designed to work offline often involve downloading large data sets during the first launch of the application (typically known as the provisioning phase of the device). The data sets can range from 30 MB to 400 MB. The initial download time can range between 10 minutes to three hours. If you do not place appropriate guards within the application, then the user can get frustrated and may abandon the application. The following are some of the guidelines to follow a good user experience:

- Ensure the user is provided with appropriate feedback regarding the data that downloads and the overall progress. You can do this using a simple progress bar as below:



  The design ensures that the user is provided with appropriate feedback on the long running activity and is well informed.

- The initial download (provisioning phase) should be performed on a WiFi network. Downloading 400 MB data on a cellular network may lead to frequent disconnects or sync errors. So you should advise users to be in an area where a good WiFi connection is available.

- You should advise users to keep their devices sufficiently charged before they perform the initial sync. Shutting down the device abruptly in middle of sync may lead to inconsistent database. In such a case, the only option is to completely delete the application and restart sync.

© 2013 by Kony, Inc. All rights reserved

3. Guidelines to Design SyncScopes and SyncObjects

While designing SyncScope and SyncObject, as a developer, you should ensure the SyncObject model represents what the application needs and is not a blind replica of the Enterprise Backend model. Do not define the attributes that you never use in the application as a part of the SyncObject. Define only those tables or business objects in the SyncConfiguration that you actually use in the application.

You should make the objects that share similar DataSource and similar synchronization characteristics as part of the same SyncScope. Synchronization characteristics include how frequently the data changes on the device and the backend.

© 2013 by Kony, Inc. All rights reserved

4. Identifying an Appropriate SyncStrategy

Deciding upon a SyncStrategy is one of the key architectural/design decisions that you need to make while developing any enterprise grade offline application. It is based on number of input parameters and system constraints. The following are some of the recommendations that can help you determine how to choose an appropriate strategy:

4.0.1 OTA Sync Strategy

OTA Sync Strategy is a recommended solution when:

- Enterprise backend is highly available to the Kony Fabric Sync Server.
- The Enterprise backend can be provisioned or unprovisioned. Provisioned means, the typical database design patterns are followed, for example, tracking deletes through a soft delete flag and tracking lastupdated timestamp for each data item (or a table row). These elements are essential for OTA Sync as they are needed for sending appropriate updates back to the client database and keep the processing as low as possible on the KonySync server.
- The Enterprise backend can provide delta changes based on the last updated timestamp the devices send.
- It is essential that the user gets access to the latest updates on the Enterprise backend every time the user performs sync.

4.0.2 PersistentSync

PersistentSync is a recommended solution when:

- The Enterprise backend is not highly available.
- A scheduled system down time does not allow user to access the Enterprise backend (due to time zone differences).
- The Enterprise backend is provisioned or unprovisioned. Provisioned is term used for signifying

© 2013 by Kony, Inc. All rights reserved

that typical database design patterns are followed, for example, tracking deletes through a soft delete flag and tracking lastupdated timestamp for each data item (or a table row). An unprovisioned backend does not follow these characteristics.

- It is acceptable that users are productive even with an information set that is "outdated" or "stale" by a few hours.

> ***Note:*** There is a need to keep the system and operational costs down by not replicating the Enterprise backend on KonySync Server.

© 2013 by Kony, Inc. All rights reserved

5. Using Filters

You may use client filters to ensure that the data downloaded to the device is optimized for the user.

Every user often deals with few thousands of records while the enterprise backend stores millions of records of all users. In such a scenario, it is important that every device has only the data the user is interested in.

Client filters enable the Kony Fabric Sync Server to filter data specific to the user before sending the response. For Database DataSource, these filters form the part of WHERE clauses and UNION queries when selecting the data for each table. For other DataSources, these are passed as input parameters to the Service Endpoints.

Similarly we have server filters. You may use server filters to filter the data coming to Sync server from backend.

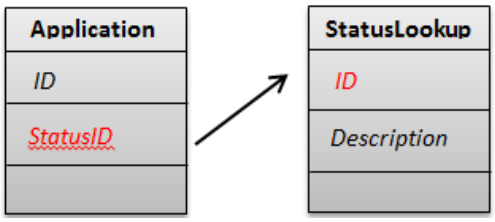
© 2013 by Kony, Inc. All rights reserved

6. Avoid Complex Relationships between SyncObjects

While defining a SyncObject model, relationships form an essential part of the definition. The user must be very careful in not replicating every relationship between the objects. You should therefore define only those relationships in SyncConfiguration that drive the application behavior.

Relationships defined in SyncConfiguration drive the primary key management done on the server as well as data filtering when downloading data to the device. The relationships also drive data creation and modification on client side.

Consider the two database tables and the relationship between them. It is clear that the StatusLookup table provides more descriptive information about a particular application status. Even though `Application` has reference to a key in *StatusLookup*, it may not be necessary to download only those StatusLookup rows as referenced by the Application. Even though this is desirable it can be over optimization. *Application* and *StatusLookup* share the parent child relationship in a real world context. So, defining a relationship between these two SyncObjects is overkill.



6.1 Differentiate between Lookup and Transactional Data

Almost in every Enterprise Backend system, there is lookup (Reference) data that changes very rarely (mostly READ) while there are transactional data that change very frequently (mostly WRITE).

By designing them as part of different SyncScopes, you can apply different synchronization characteristics. You can synchronize lookup data every two days while you can refresh the transactional data every two hours.

Lookup data is often same across users, and is maintained in a single DataSource at the backend. To differentiate the data specific to a user you can apply a client filter.

© 2013 by Kony, Inc. All rights reserved

6.2 Identify Appropriate Batch Size for the Application

When downloading data to the device, the Kony Fabric Sync Server does not send all the data in one response. It sends them in batches. The user defines the batch size while initiating a sync (default batch size is 500).

As you have batchsize for download, similarly, you have batching for upload as well . You can break the data to upload into batches to reduce the load on network. To achieve this, you need to configure "uploadbatchsize" config param for *sync start*.

A very small value of batch size results in many network calls and hence can lead to poor sync performance. A very large value of batch size can lead to Out Of Memory issues on the server and the device (as all the data in batch is loaded in memory).

You are recommended to perform some tests (using different batch sizes) with real world application data to arrive at the batch size that is more appropriate for the application.

> ***Note:*** Batching does not work as is, if there is huge data that is almost similar to the value of the timestamp.

6.3 Getting the SyncObject Relationships Right

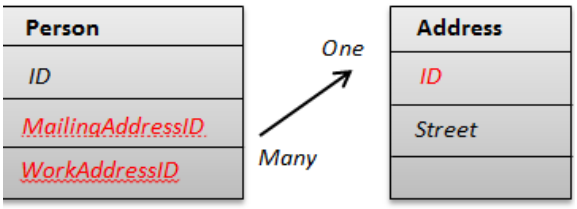
You need to get the SyncObject relationships right for Database datasource.

Kony Fabric Sync currently supports *One-To-Many* and *Many-To-One* relationships between the SyncObjects. It is important to understand how the developer should decide which side of the relationship is the Many side. The below example illustrates how to identify that.

The *Many* side is always defined on the table that has the Foreign Keys.

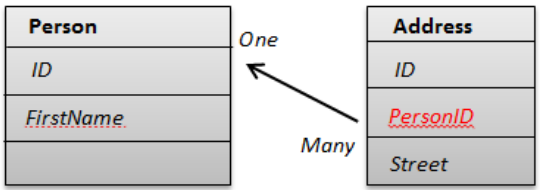
Example 1

© 2013 by Kony, Inc. All rights reserved



In the above example, since the Foreign keys exist on the `Person` table it becomes the Many side of the One-To-Many or Many-To-One relationship.

Example 2



In the above example, since the Foreign keys exist on the `Address` table it becomes the *Many* side of the One-To-Many or Many-To-One relationship.

It is equally important to identify where to define this relationship. Let us say in *Example 1*, we can define the *Many-To-One* relationship on the `Person` SyncObject or One-To-Many relationship on the `Address` SyncObject. The relationship should be defined on the SyncObject whose filters have to be inherited by the other Syncobject. For example; if the application demands that only selected PersonIDs that the user defined should be downloaded (so a client filter on `Person`) and only those Address records that the client filter selected, need to be downloaded, then you have to define the relationship on the `Person` SyncObject. (As in this case `Address` inherits the filter criteria from `Person`).

6.4 Define Appropriate Indexes

In case of database datasource, ensure that appropriate indexes are defined for all the columns that are part of the relationship and sync filters (both client and server).

> ***Note:*** Indexes can affect the WRITE performance on the tables.

© 2013 by Kony, Inc. All rights reserved

6.5 Optimize the SOAP Requests

Let us consider SalesForce Datasource as an example to optimize SOAP requests.

SalesForce offers a very rich query interface through its WebService interface. This API allows a user to define its own objects as well as construct complex queries to select only the required fields.

With respect to SalesForce, you must not download deleted records to the device when downloading data for the initial downloads. You can do this by having a code snippet similar to below, in the SOAP request template for *getAll* operation mapping.

```
<urn:queryString>select Id, IsDeleted, MasterRecordId, Name, Type,
RecordTypeId, LastModifiedDate from Account where
LastModifiedDate&gt;$StartDate and LastModifiedDate&lt;$EndDate #if
($InitialSync=='true') and IsDeleted=false #end</urn:queryString>
```

SFDC expects a specific NULL tag to set the value of an attribute to null. So one must modify the **update** and **create** SOAP request template in the operation mapping.

```
#if($Name != "null")
<urn1:Name>$Name</urn1:Name>
#else
<urn1:fieldsToNull>Name</urn1:fieldsToNull>
#end
```

© 2013 by Kony, Inc. All rights reserved

7. Guidelines for Web Services

In this section you will find guidelines to implement web services and the SAP DataSource connector to integrate with Kony Fabric Sync Server.

7.1 Services

Kony Fabric Sync Framework operates on SyncObjects (for example, Contact and Product). To read or update data in these objects, Kony Fabric Sync Framework expects the Enterprise backend to provide a CRUD (Create, Read, Update, Delete) friendly interface. This helps the Kony Fabric Sync Framework to map actions performed on the data to be replicated on the Enterprise backend. The web service should have the following operations defined for each SyncObject.

> ***Note:*** There is no particular naming convention on Service name, and the request and response field names. For example, service name can be “Contact_Create” or “ContactCreate” or “ContactInsert” for the Create operation.

Operation	Description
Create	Service that takes all the field level data in the request and creates an instance in the enterprise data source and returns the auto generated primary key information in response (if primary key is auto generated).
Update	Service that takes all the field level data along with primary key in the request and updates the instance in enterprise data source.
Delete	Service that takes the primary key of the row and soft-deletes the row from the data source.

© 2013 by Kony, Inc. All rights reserved

Operation	Description
Get	Service that takes the primary key of the row in the input and gives all the field data in the response. This service is typically used for doing conflict resolution. This is also invoked when GetUpdated returns only primary keys and does not return all the data values for a SyncObject. In case of some services, it may not take any inputs and return complete data always. In such scenarios, the service does not perform delta sync.
GetUpdated	Services that takes the date time value in the request and returns all the rows (with all the field data) changed after that date time.
GetBatch	Service that provides ability to retrieve data in batches.
GetServerTime	This is defined once at datasource level. It returns current datetime value of the datasource. It is not mandatory to define this service, but you are recommended to have this service to ensure that a very precise batching is done for large datasets. So consider this while designing the services.

> ***Note:*** There is no restriction on the name of the WebService methods but the semantics/behaviour has to be similar to what is described above. There is "getUpdated" operation for OTA and persistent provisioned sync strategy, and the "getUpdated" operation is renamed to "GetAll" for persistent unprovisioned sync strategy.

See *Datasources > Other DataSources* section in Framework Getting Started guide on Kony Documentation Library to understand more about the service requirements and batching support.

7.1.1 Mapping Input and Output Parameters for Operations

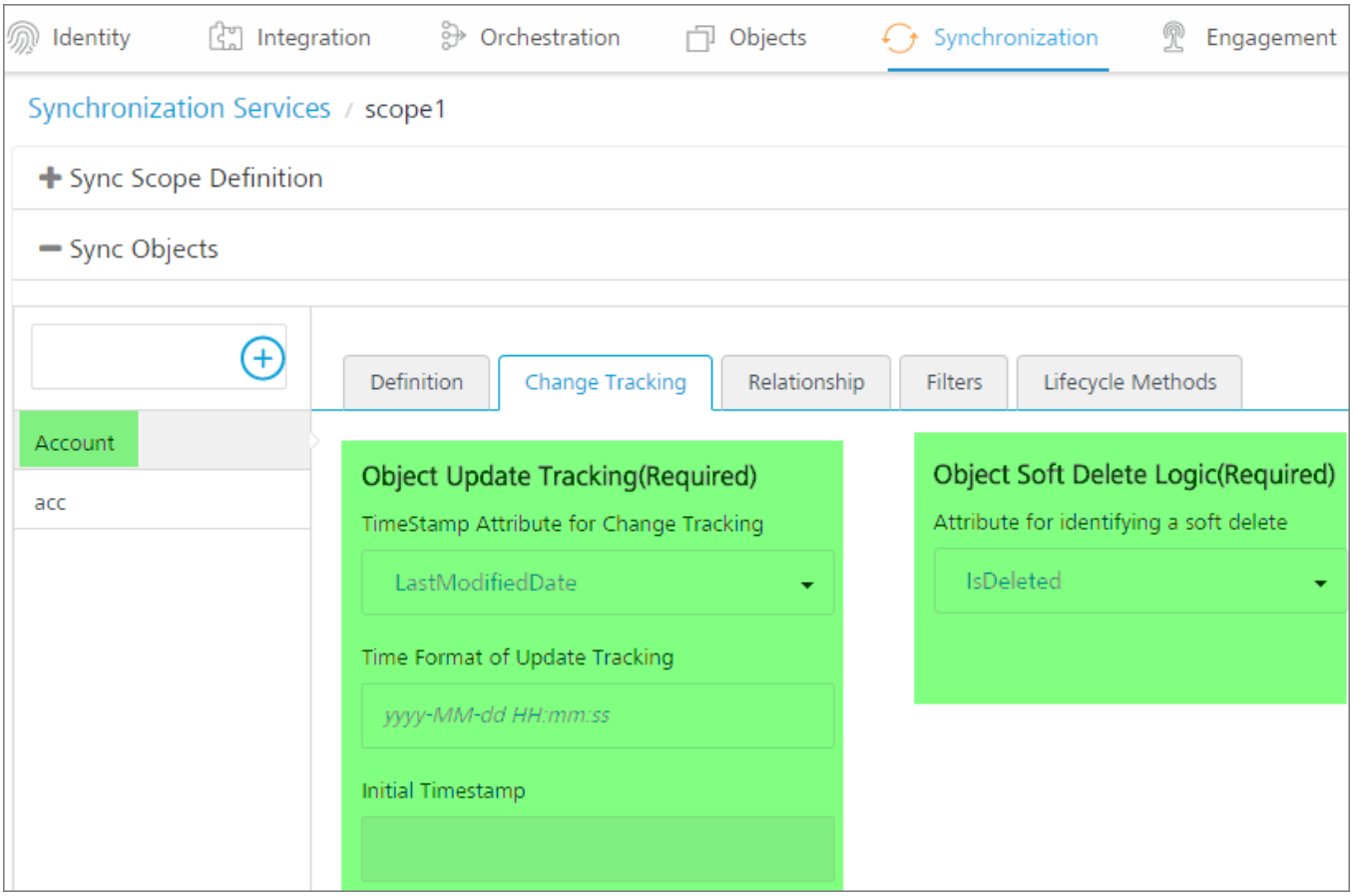
Input parameters are the parameters that you pass to the service and are used during a service call. All the services do not have input parameters. Input parameters are provided based on the elements that are present in the Request pane.

© 2013 by Kony, Inc. All rights reserved

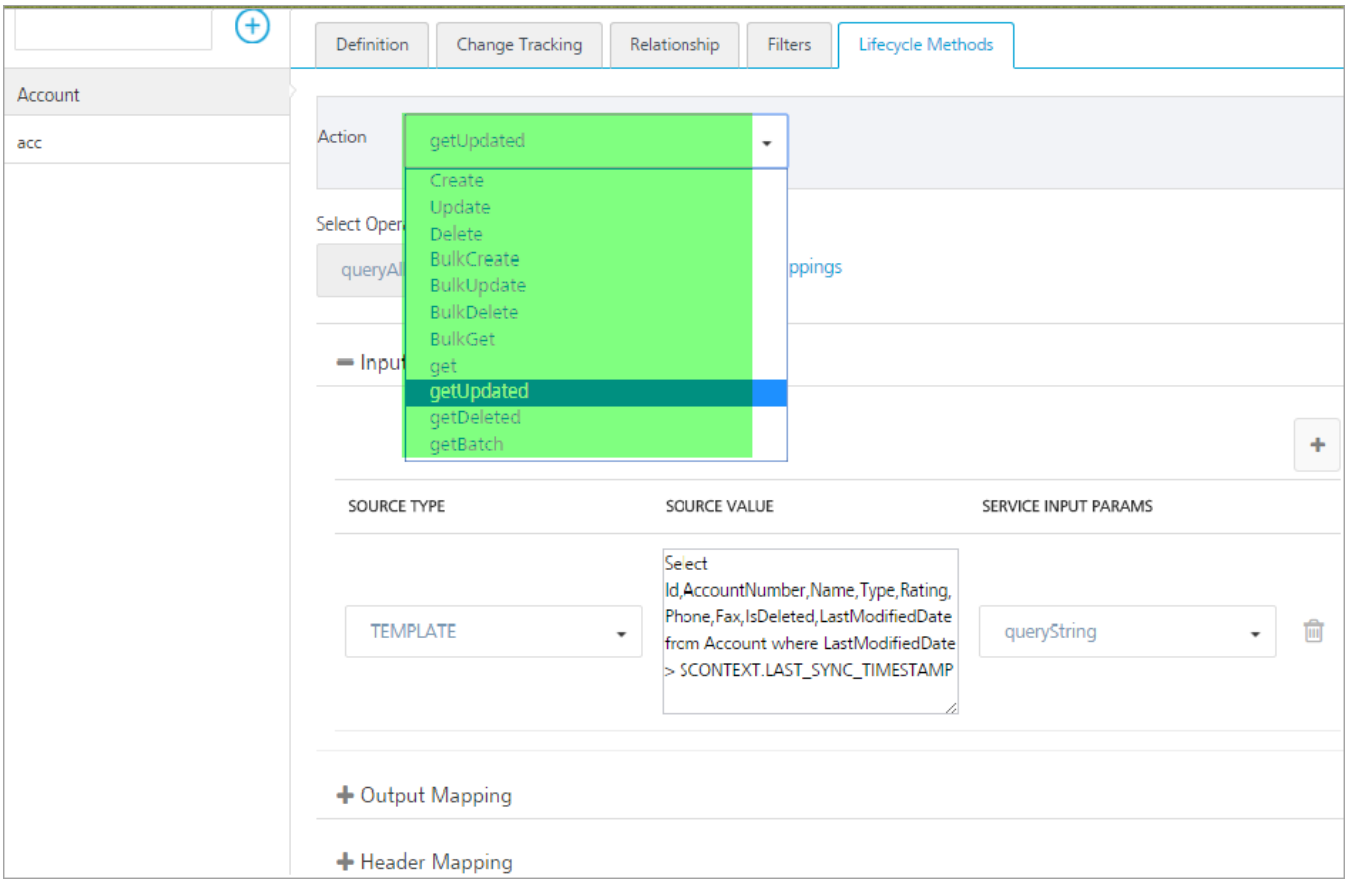
> ***Note:*** You have to map all the input parameters expected by the service in Input mapping.

Output parameters are the parameters fetched from the response of a service call.

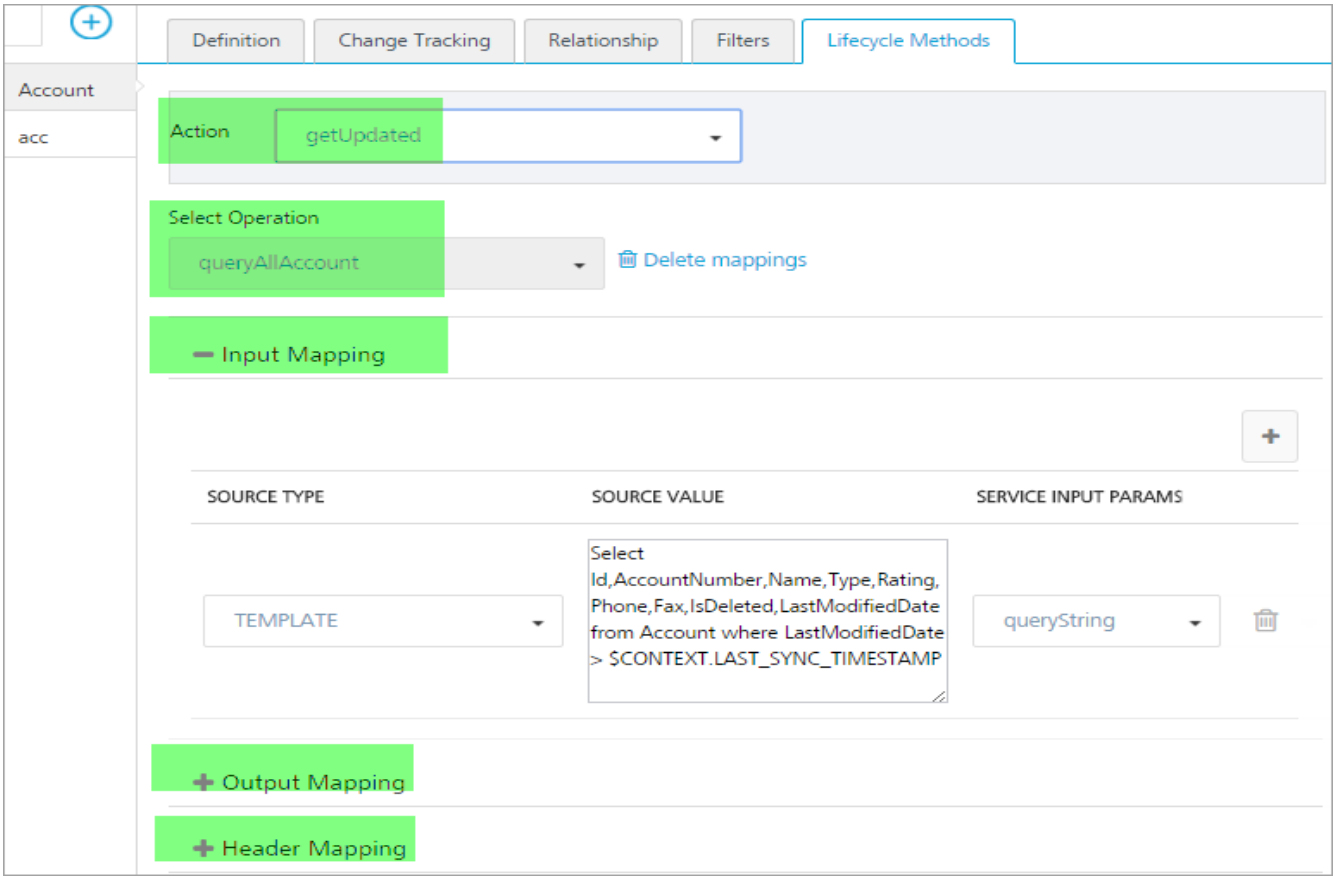
Example: The screen shots below will help you understand mapping parameters. The example shows a Contact object based on the Salesforce model.



© 2013 by Kony, Inc. All rights reserved



© 2013 by Kony, Inc. All rights reserved



Ensure that the following guidelines are adhered to while mapping the input and output parameters:

Create Input Mapping

This operation has to accept all or subsets of fields from Kony Fabric Sync attributes of the same object as well as from other objects, if a relationship exists between the objects.

Create Output Mapping

If there is an auto-generated key, then you have to map the auto-generated value in output mapping.

Header Mapping

Header Mapping is done for all the operations.

Update Input Mapping

© 2013 by Kony, Inc. All rights reserved

This operation has to accept all or subsets of fields from the attributes. This operation has to accept all or subsets of fields from Kony Fabric Sync attributes of the same object as well as from other objects, if a relationship exists between the objects.

Update Output Mapping

This operation does not require output mapping. If the service fails with SOAP fault or with failed HTTP status code, it implies that the update has failed. Otherwise, it is considered as successful.

Delete Input Mapping

This operation has to accept the primary key of the row.

Delete Output Mapping

This operation does not require output mapping. If the service fails with SOAP fault or with failed HTTP status code, it implies that the update has failed. Otherwise, it is considered as successful.

Get Input Mapping

This operation has to accept the primary key of the row.

Get Output Mapping

This operation should return all the field level data for that particular primary key.

GetUpdated Input Mapping

This operation can accept the last sync time (lower bound) and current timestamp (upper bound), and return the rows that are added, changed, or deleted between the lower and upper bound values.

GetUpdated Output Mapping

This operation should return the list of rows containing all the field level data. For batching support, the service has to return one parameter flag that indicates whether more batches are available. The service can also return identifiers like QueryLocator that can be stored in context and can be sent as GetBatch Input.

© 2013 by Kony, Inc. All rights reserved

GetBatch Input Mapping

The GetBatch operation will be invoked when the moreChangesAvailable flag from the GetUpdated operation is true. This operation will be invoked by Kony Fabric Sync Server recursively until the flag is false.

GetBatch Output Mapping

The output should be the same as GetUpdated operation output.

For batching, the operation takes batch size as an input parameter and returns the batch size number of rows. The number of rows in the output can differ from batch size.

7.2 Provisional Columns

Every SyncObject should have defined provisional columns in addition to the service operations. Provisional is a term that signifies typical database design patterns have been followed, such as tracking deletes through a soft delete flag and tracking last updated timestamp for each data item or a table row.

The Change Tracking Columns (for example, `Last UpdateTimestamp` and `Soft Delete Flag`) must be defined for each SyncObject.

SoftDeleteFlag

This column represents whether the row is deleted. The column captures a Boolean flag (true/false) that is used to track if the row has been marked for deletion. This column is optional if the SyncObject does not deal in delete functionality.

LastUpdateTime

This column indicates the time when the row is last modified.

Error handling

© 2013 by Kony, Inc. All rights reserved

If the service is invoked and there are errors, Kony Fabric Sync Server will handle the error depending on the web services (SOAP, REST and JSON). Kony Fabric Sync Server will also notify the device. For example, in case of SOAP service, SOAP Fault is considered as an exception. For REST and JSON services, errors will be handled based on the HTTP status codes.

For handling custom errors from the backend services, map one service output parameter with name `errmsg` with XPath expression to the error message in response payload to catch the exact error.

If the web service is the DataSource and errors are reported in SOAP response instead of SOAP Faults, then add the below param in the `Synconfig` file under each service output param to catch the exact error. For example, to use salesforce web service, add as shown:

```
<Param Name="errmsg" Expression="//errors/message/text()"
Datatype="string"/>
```

Refer Developing Offline Apps on Kony Documentation Library for more information.

7.3 Batching Approach

You must use the GetUpdated and GetBatch operations to retrieve data in batches. If you do not have the service already implemented for the batching purpose, then you can implement the service for each SyncObject by following the algorithm as shown to fetch the batch-size number of rows from the backend DataSource. After implementing the service, map the service with GetUpdated and GetBatch operations. Kony Fabric Sync Server will invoke this service to get the incremental changes in batches. Data will be sent to a device in batches to avoid memory and network issues when there are more rows in initial sync and incremental sync.

Algorithm to fetch the batch-size number of rows from the backend DataSource

```
Algorithm <entity> GetUpdated(datetime, batchSize)
cursor ← Open a cursor on <entity> where lastUpdateTime > datetime
recordCount ← 0
prevLastUpdateTime ← 0
row ← cursor.firstrow()
while recordCount < batchSize or row.lastUpdateTime ==
```

© 2013 by Kony, Inc. All rights reserved

```
prevLastUpdateTime do
 recordCount ← recordCount + 1
 prevLastUpdateTime ← row.lastUpdateTime
 list.add(row)
 row ← cursor..nextrow()
end-while
lastRowDatetime ← row.lastUpdateTime
cursor.close()
if recordCount >= batchSize then
 moreDataAvailable ← true
else
 moreDataAvailable ← false
end-if
Return list, moreDataAvailable, lastRowDatetime
```

Operation mapping

- In GetUpdated input mapping, map the following:
  - `LAST_SYNC_TIMESTAMP` from the context to `datetime`
  - `BATCH_SIZE` from context to `batchSize`
- In GetUpdated output mapping, map the fields from `list` to the corresponding Kony Fabric Sync attributes as shown:
  - `moreDataAvailable` to `MORE_CHANGES_AVAILABLE` in context
  - `lastRowDatetime` to any temporary variable in context (for example, QueryLocator)
- In GetBatch input mapping, map the following:
  - Temporary context `variable(QueryLocator)` from the context to `datetime`
  - `BATCH_SIZE` from context to `batchSize`

© 2013 by Kony, Inc. All rights reserved

- The GetBatch operation output mapping will be the same as GetUpdated operation output mapping.

© 2013 by Kony, Inc. All rights reserved

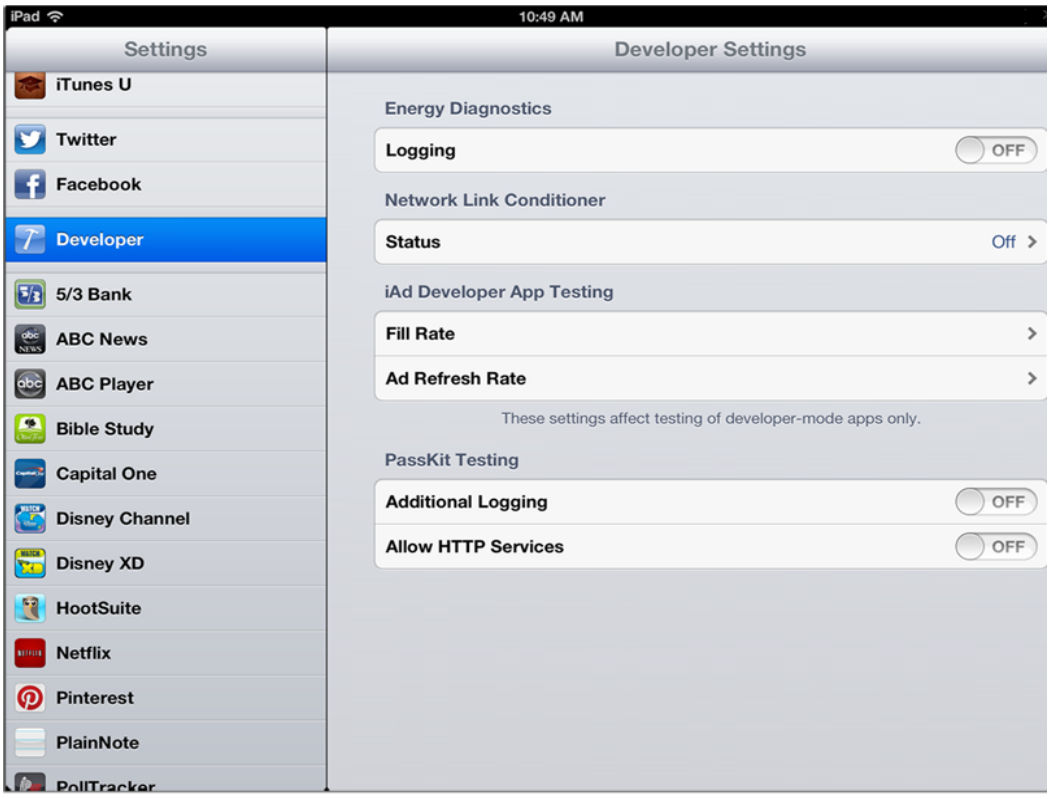
8. Simulating and Testing for Different Network Conditions

Being able to test different network conditions is important for an application that downloads or uploads large amount of data. It is important to understand the user experience under such conditions. The last thing you want is the user being stuck, or some functionality not working at all because the user is on a mediocre networking connection.

One way to simulate a bad network connection on iOS is to use tools like “Network Link Conditioner”.

Refer the below URL for a detailed usage of “Network link Conditioner”:

http://www.neglectedpotential.com/2012/09/ios6-network-link-conditioner/



Another tool useful for simulating bad network connections is “Charles Proxy” http://www.charlesproxy.com/. It allows you to record traffic as well simulate slow network connections.

© 2013 by Kony, Inc. All rights reserved

9. Load Testing Kony Fabric Sync Services with Apache JMeter

This section explains the steps required to record Kony Fabric Sync service calls with Apache JMeter. If you are new to JMeter, create a test plan by recording Kony Fabric Sync service calls with JMeter using Android emulator.

Download and install JMeter from http://jmeter.apache.org/download_jmeter.cgi

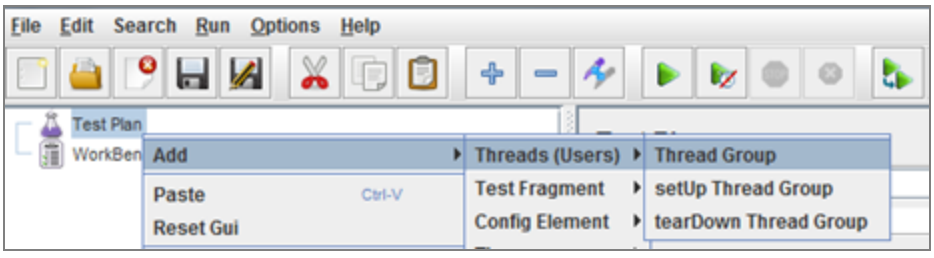
Prerequisite

- Kony Fabric Sync Server
- Kony Fabric Sync Application must be synched with data resources to get data.

> ***Note:*** This document does not explain how to use JMeter in detail. For more information on using JMeter, refer to http://jmeter.apache.org/usermanual/
> Make sure Kony Fabric Sync Server is running and the Kony Fabric Sync application is reset.

To perform load testing on Kony Fabric Sync Service calls, do the following:

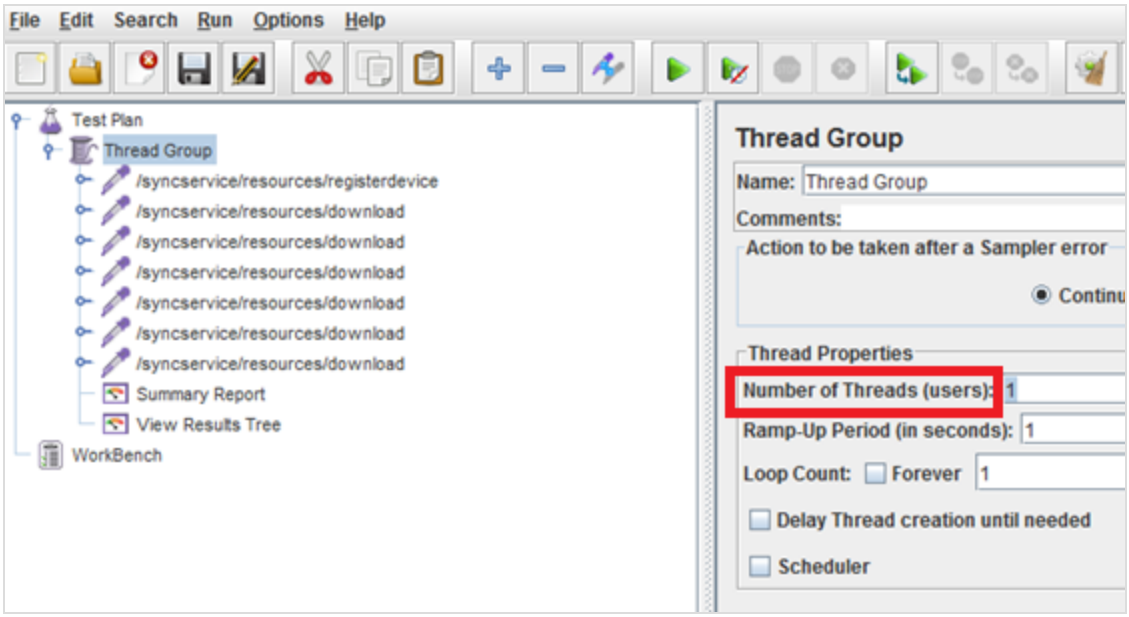
1. Build a basic test plan in JMeter
    1. Navigate to **JMETER_HOME/bin**.
    2. Launch **jmeter.bat**. JMeter window appears.
    3. Right-click on **Test Plan** and then select **Add** > **Threads (Users)** > **Thread Group**. Thread group is added to test plan.



© 2013 by Kony, Inc. All rights reserved

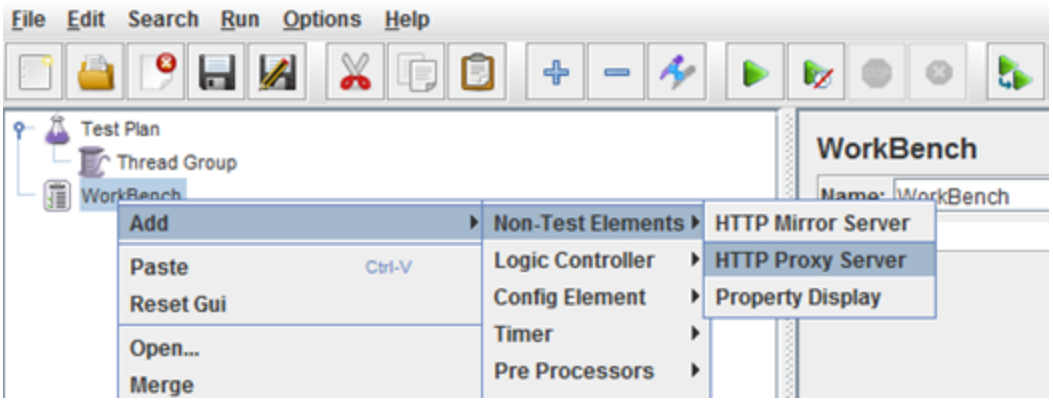
4. Click **Thread Group**. Thread group window appears.

5. Under **Thread Properties**, enter the number of threads or users you want to view.



2. Add a HTTP Proxy Server.

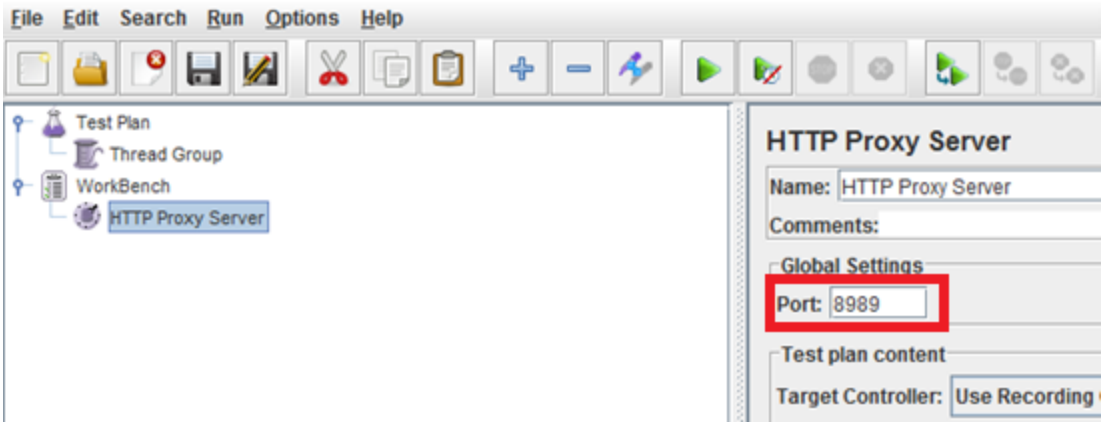
1. In JMeter, right-click on **WorkBench** and then select **Add** > **Non-Test elements** > **HTTP Proxy Server**. HTTP proxy server is added to WorkBench.



2. Click HTTP Proxy Server. HTTP Proxy Server details appear.

© 2013 by Kony, Inc. All rights reserved

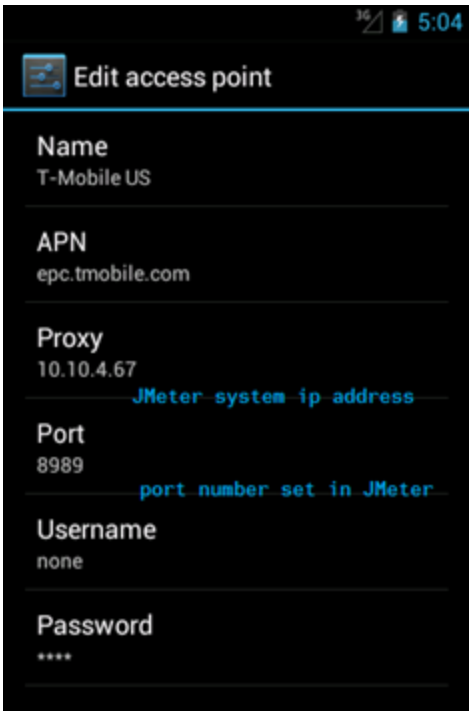
3. Under **Global Settings**, enter **Port** number.



3. Set up Android emulator.
    1. From Kony Visualizer, launch Android emulator.
    2. Navigate to **Settings** > **Wireless & Networks** > **Mobile Networks** > **Access Point Names**.
    3. Select default access point name, **T-Mobile US**.

© 2013 by Kony, Inc. All rights reserved

4. In the T-Mobile US proxy, edit the following.
    1. **Proxy**: Set the IP address of the system where JMeter is running.
    2. **Port**: Enter the port number specified in JMeter.



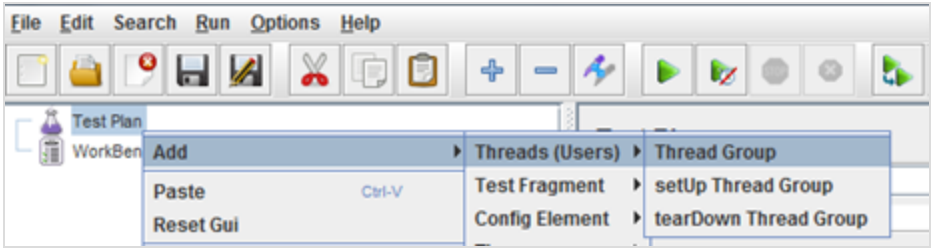
4. Launch Kony Fabric Sync application in the Android emulator and start the Kony Fabric Sync application.
5. In JMeter, click **Start** to start load test.

9.1 Recording Kony Fabric Sync Requests of a Windows 7 Desktop Application with JMeter

To record Kony Fabric Sync requests of a Windows 7 desktop application with JMeter, follow these steps:

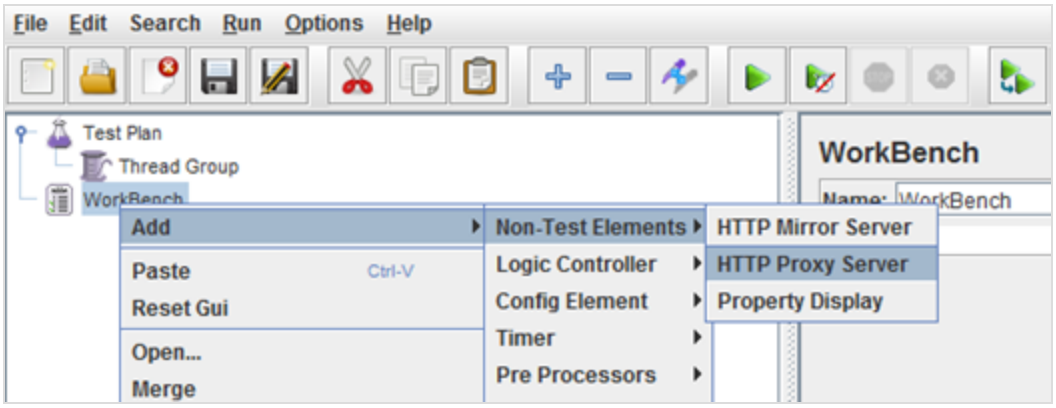
© 2013 by Kony, Inc. All rights reserved

1. In your Internet Explorer, from the **Tools** menu, click **Internet Options**. The Internet Options window appears.
2. Click on **Connections** tab.
   The Connections tab details appear.
3. Click **LAN Settings**
   The Local Area Network settings window appears.
4. Under **Proxy Server**, select Use a proxy server for LAN.
5. Click **Advanced**.
   The **Proxy Settings** window appears.
6. Enter the following details
   1. **HTTP**: Enter HTTP server details.
   2. **Port**: Enter port number.
7. Navigate to **JMETER_HOME/bin**.
8. Launch `jmeter.bat`. JMeter window appears.
9. Right click on **Test Plan** and then select **Add** > **Threads (Users)** > **Thread Group**. Thread group is added to test plan.



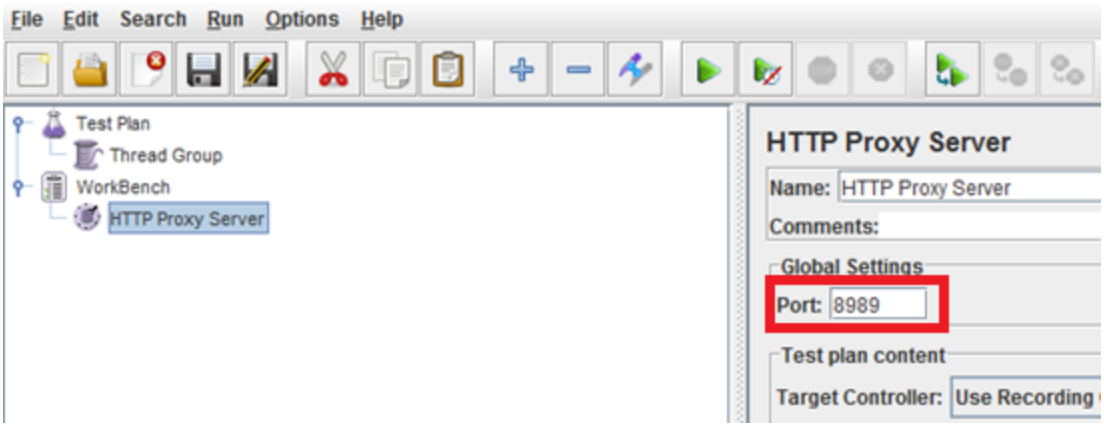
10. In JMeter, right-click on **WorkBench** and then select **Add** > **Non-Test elements** > **HTTP Proxy Server**. HTTP proxy server is added to WorkBench.

© 2013 by Kony, Inc. All rights reserved



11. Click **HTTP Proxy Server**. HTTP Proxy Server details appear.

12. Under **Global Settings**, enter **Port** number. This is the same number you have provided in your Internet Explorer proxy settings.



13. In JMeter, click **Start** to start load test.

> ***Note:*** To compare the load test results, refer to **Monitor and Tune Kony Fabric Sync Server Performance**section in *Server Planning Guide* on Kony Documentation Library.

© 2013 by Kony, Inc. All rights reserved