SyncML Manual Test Cases, version 0.5

Abstract

These manual test cases are defined by the SyncML Interoperability Committee for use within the SyncML Conformance Testing process.

Consortium

The following companies are sponsors in the SyncML initiative:
Ericsson
IBM
Lotus
Matsushita Communications Industrial Co.
Motorola
Nokia
Palm, Inc.
Psion
Starfish Software

Editor:

Peter Thompson (Starfish), mailto:peter.thompson@starfish.com

Rob Scott (IBM), mailto:robhs@us.ibm.com

Revision History

Revision	Date	Comments
0.1	2000-09-20	First draft
0.2	2001-01-25	Changes according to TechCom review in SV
0.3	2001-01-29	English check
0.4	2001-02-02	Text enhancement
0.5	2001-02-05	Minor textual changes

Copyright Notice

Table of Contents

1 Introduction

These manual test cases are defined by the SyncML Interoperability Committee (SIC) for use within the SyncML Conformance Testing Process [1]. Until test software is available, these test cases will form the basis of the conformance testing. These test cases may be updated from time to time by the SIC. Publication of revisions of this document before the end of February 2001 will be announced via the SyncML Supporter mailing list, and the revised version will be posted on the SyncML web site www.syncml.org/interop/index.html.

The goals of these manual test cases are:

- Prove comprehensive coverage of required functionality for both clients and servers.
- Prove that the device is sending and receiving valid SyncML messages.
- Prove that the device acts according to the SyncML Protocol.

2 Using this Document

After the test cases have been run, this document should be completed by the vendor and returned to the SIC for review, along with a completed SyncML Implementation Conformance Statement Proforma [2]. Electronic submission of test reports will be accepted.

3 Vendor Data

3.1 Product and Vendor Contact Information

This section must be completed by the vendor applying for SyncML conformance. The product name is simply an identifier, it does not have to be the public name of the product.

Product Name & Version	
Company	
Contact Name	
Contact Phone	
Contact Email	
Transports supported	OBEX [ ] WSP [ ] HTTP [ ]
Product is	Client [ ] Server [ ]
Data types product uses	

3.2 Test Results

The vendor must fill out the following table with the results of the tests. The Pass? column must be filled out YES or NO. If the vendor answers NO, then they must fill in the Comment column.

Test #	Pass?	Comment
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		

4 Manual Test Cases

Each test case presented here starts with a title and a brief description of the test case. The setup describes any necessary steps prior to actually running the test case. In some of the cases, it will be necessary to view either log files or captured SyncML data to prove the test case worked properly.

4.1 Test Case #1: Initial Two-way Sync

4.1.1 Purpose

This test shows basic compliance with the SyncML protocol and representation through the first-time use of a two-way synchronization. The intent is to show the exchange of Device Information as well as an empty two-way synchronization. The test should also show proper implementation of the Sync command when there is no data to send.

4.1.2 Setup

Clear the client and server databases – this will ensure that no data other than Device Information is exchanged.
Using viewers, verify that the client and the server contain no records.
Create any appropriate synchronization accounts on the client and server.

4.1.3 Actions

Perform a two-way synchronization.
Using viewers, verify the two-way sync actually took place. This can be done by viewing captured SyncML data or by viewing log files.

4.1.4 Expected Results

The client and the server should contain no records.
The client and server should have exchanged device information.
The synchronization should have produced no errors.

4.1.5 Possible Errors

Synchronization may fail on transport layer.
Synchronization may fail on protocol layer.
Synchronization may fail on authorization.
Synchronization may fail on device info.

4.2 Test Case #2: Two-way Sync with Client and Server Add command

4.2.1 Purpose

This test shows the implementation of the Sync command with data and the Add commands. On simpler devices, Replace can be used instead of the Add command.

4.2.2 Setup

This test requires successful completion of Test Case #1.
This test also requires agreement between the client and server on which objects are to be synchronized (e.g. vCard 2.1).

4.2.3 Actions

Create a new object on both the client and the server, filling in as many fields as possible.
Perform a two-way sync.
Using viewers, verify both objects exist correctly on the client and the server.

4.2.4 Expected Results

Both client and server will contain both objects.
Server should contain all the fields set in the object created on the client.

4.2.5 Possible Errors
Objects may not appear on the client or the server – this would indicate the Add failed for some reason.
Required fields may not be filled in on client or server – see object conformance tables.
Data may appear in wrong fields – this would indicate improper implementation of the object.

4.3 Test Case #3: Two-way Sync with Client and Server Replace command

4.3.1 Purpose
This test shows implementation of the Replace command, as well as implementation of the ID mapping capability.

4.3.2 Setup
This test requires successful completion of Test Case #2.

4.3.3 Actions
On the client, modify as many fields as possible in the object created on the server.
On the server, modify as many fields as possible in the object created on the client.
Perform a two-way sync.
Using viewers, verify both objects have been updated correctly on client and server.

4.3.4 Expected Results
Both objects should have all the modified fields updated on the client and the server.

4.3.5 Possible Errors
Object modified on client not updated on server – this may be due to Replace failing or MapItem not implemented properly, or mapping implemented incorrectly.
Object modified on server not updated on client – this may be due Replace failing, but not mapping implemented incorrectly.

4.4 Test Case #4: Two-way Sync with Client and Server Delete command

4.4.1 Purpose
This test shows implementation of the Delete command, as well as implementation of the ID mapping capability.

4.4.2 Setup
This test requires successful completion of Test Case #2 or #3.

4.4.3 Actions
On the client, delete the object created on the server.
On the server, delete the object created on the client.
Perform a two-way sync.
Using viewers, verify both objects have been deleted from the client and the server.

4.4.4 Expected Results
Both objects should be deleted from the client and the server.

4.4.5 Possible Errors
Object deleted on client not deleted on server – this may be due to Delete failing or MapItem not implemented properly.
Object deleted on server not deleted on client – this may be due Delete failing.

4.5 Test Case #5: Two-way Sync with Client Add command

4.5.1 Purpose
This test shows two-way sync with server sending no sync data. This will require the sending and handling of an empty Sync command. It is possible that problems may arise from an empty Sync command being sent as well as a non-empty Sync.

4.5.2 Setup
This test requires a previously successful two-way sync.

4.5.3 Actions
Create a new object on the client.
Fill in as many fields as possible.
Perform a two-way sync.
Using viewers, verify that object exists correctly on the client and the server.

4.5.4 Expected Results
Object should exist on both client and server.

4.5.5 Possible Errors
Object may not appear on the server.
Required fields may not be filled in on client or server – see object conformance tables.
Sync may fail due to no data from client .

4.6 Test Case #6: Two-way Sync with Server Add command

4.6.1 Purpose
This test shows two-way sync with client sending no sync data. This will require the sending and handling of an empty Sync command. It is possible that problems may arise from an empty Sync command being sent as well as a non-empty Sync.

4.6.2 Setup
This test requires a previously successful two-way sync.

4.6.3 Actions
Create a new object on the server, filling in as many fields as possible.
Perform a two-way sync.
Using viewers, verify that object exists correctly on the client and the server.

4.6.4 Expected Results
The new Object should exist on both client and server.

4.6.5 Possible Errors
The Object may not appear on the client – this would indicate the Add failed.
Required fields may not be filled in on client or server – see object conformance tables.
The Sync may fail due to no data in the Sync command from server – the client may fail to handle the empty Sync command.

4.7 Test Case #7: Two-way Sync with Large number of Objects

4.7.1 Purpose
This test shows server implementation of multiple messages. This test also shows the client's ability to handle multiple messages. If necessary, it will also show the client rejecting messages due to out of memory problems. Note that it may not be possible to show this for some clients.

4.7.2 Setup
This test requires a previously successful two-way sync.

4.7.3 Actions
Create enough Objects on the server to force at least two messages to the client, but not so much as to have the client run out of space.
Perform a two-way sync.
Using viewers, verify that the objects exist correctly on the client and the server.
Again, using viewers, show that multiple messages were sent.

4.7.4 Expected Results
Objects should exist on both the client and the server.
Multiple messages should have been sent by the server.
Multiple messages should have been handled by the client.

4.7.5 Possible Errors
Not all objects may appear on the client.
Required fields may not be filled in on client – the client may not implement object conformance.
Sync was rejected due to single large message sent – the server may not have sent multiple messages.
Crashing due to overloading – the client or server may crash from mishandling out-of-memory problems.

4.8 Test Case #8: Two-way Sync with Large number of Objects Returned

4.8.1 Purpose
This test shows client implementation of sending multiple messages. This test shows the client's ability to create multiple messages to the server. It also shows the server ability to respond properly to multiple messages from the client.

4.8.2 Setup
This test requires successful completion of Test Case #7.

4.8.3 Actions
Modify all objects sent by the server in Test Case #7 (to force at least two messages to the server).
Perform a two-way sync.
Using viewers, verify that the objects exist correctly on the client and the server – this will require viewing each object and verifying the data is correct.

4.8.4 Expected Results
All Objects should exist on both the client and the server, and their data should match.

4.8.5 Possible Errors
Not all objects may not appear on the client or the server – it is possible some were deleted or not added.
Required fields may not be filled in on the server – the server may not implement object conformance.
Sync was rejected due to single large message sent – the client may have tried to send a single message.
Crashing due to overloading – the client or server may crash from mishandling out-of-memory problems.

4.9 Test Case #9: Two-way Sync with Server responding as busy

4.9.1 Purpose
This test shows ability of the client to deal with busy alert.

4.9.2 Setup
Somehow make the server be able to respond as busy.

4.9.3 Actions
Perform a two-way sync.

4.9.4 Expected Results
Client and server logs should report that the server is busy. It is also possible that the user may be prompted to try again at a later time.

4.9.5 Possible Errors
Client not understanding busy alert.
Server not sending busy alert, just default error.

4.10 Test Case #10: Two-way Sync with Server not responding

4.10.1 Purpose
This test shows client ability to deal with network or server error.

4.10.2 Setup
The server must be disabled, or set to not respond.

4.10.3 Actions
Perform a two-way sync.

4.10.4 Expected Results
The client log should report that the server is not responding.
The user may be prompted that server is not available.

4.10.5 Possible Errors
The client may wait indefinitely – this would indicate the timeouts are not dealt with properly.
The client may crash – this would be due to attempting to read bad data.

4.11 Test Case #11: Two-way Sync with incomplete communication

4.11.1 Purpose
This test shows client and server ability to deal with incomplete communication during a session.

4.11.2 Setup
This test requires a previously successful two-way sync.

4.11.3 Actions
Create a new object on both the client and the server, filling in as many fields as possible.
Perform a two-way sync.
Break communication during sync. Depending on the device, it could be as simple as unplugging the client from the network to as complex as putting the device into a Faraday cage.
Wait for sync to finish.
Perform two-way sync.

4.11.4 Expected Results
Both client and server logs should report that the session was interrupted before completion.
The second sync should be successful. (Both objects synced properly.)

4.11.5 Possible Errors
Both objects not synced after second attempt – one of the devices improperly marked the object as synchronized.
Object data mangled – one of the devices attempted to interpret bad data.
Client or server may crash – one of the devices attempted to interpret bad data.

4.12 Test Case #12: Two-way Slow Sync

4.12.1 Purpose
This test shows implementation of slow sync.

4.12.2 Setup
This test requires a previously successful two-way sync.

4.12.3 Actions
Configure the server and/or the client to ask for Slow-Sync.
Perform a two-way sync.
Verify that slow sync took place.

4.12.4 Expected Results
Slow-Sync should be indicated in the log of the client and server.
User may be prompted that the session will be a slow sync.

4.12.5 Possible Errors
Slow-Sync not performed – either the client or the server rejected the Slow-Sync request.
Slow-Sync alert not accepted – either the client or the server rejected the Slow-Sync request.

5 References

[1] SyncML Conformance Testing Process

[2] SyncML Implementation Conformance Statement Proforma