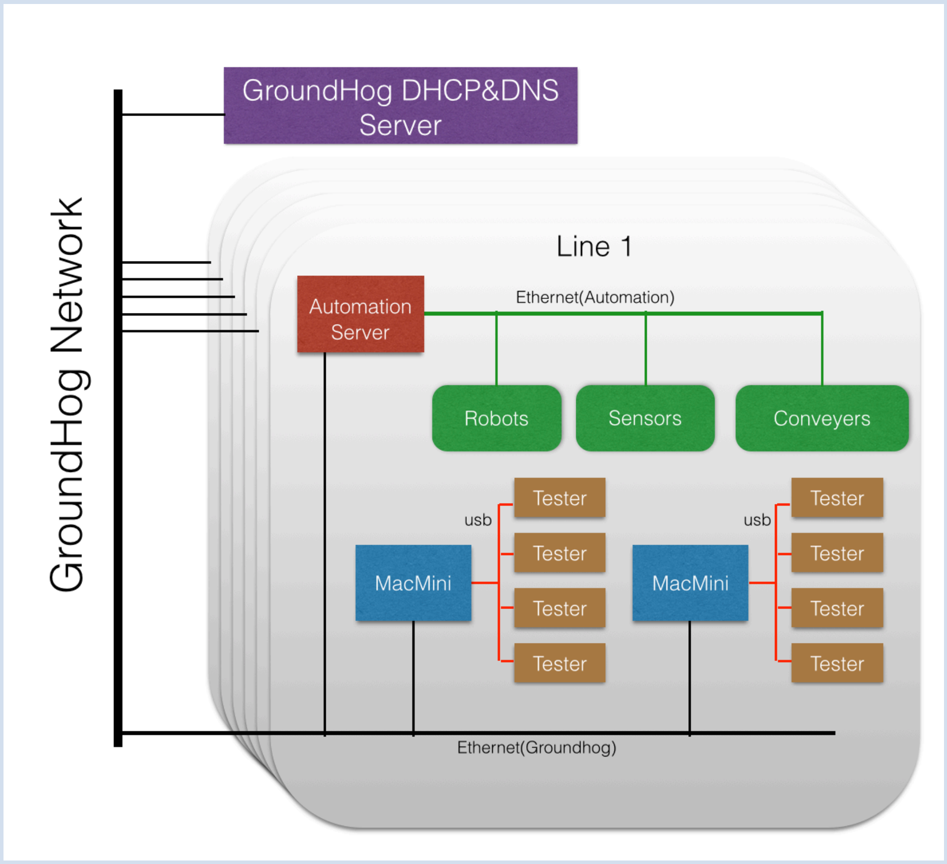
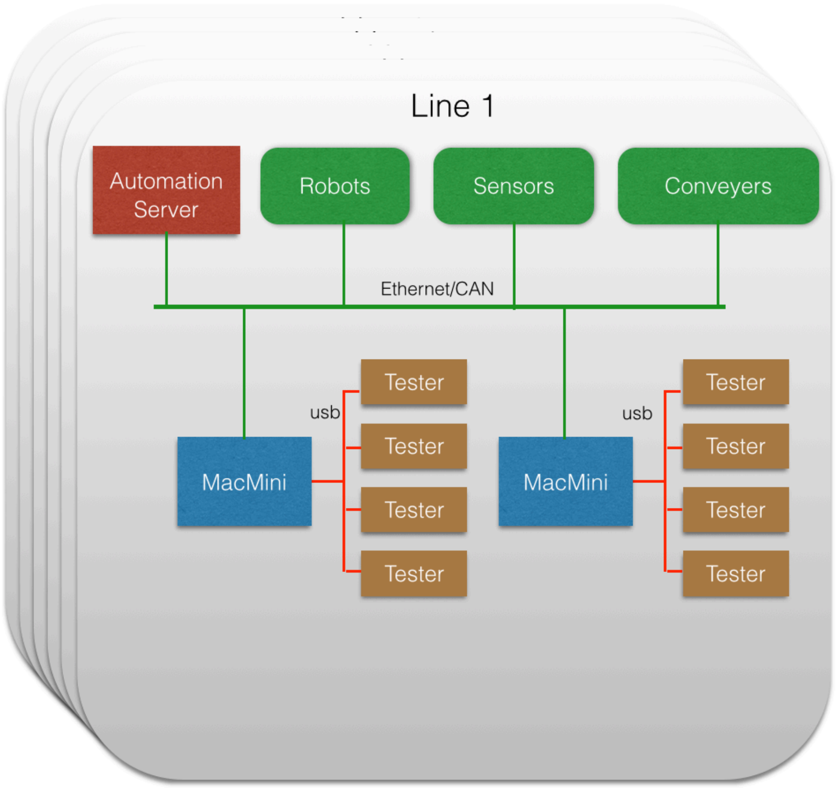
PPO-Test Automation SW Interface Specification

Apple PPO-Test

July 6, 2016

Table of Contents

Is Requirement	Section Number	Section Content
	1	**Introduction**
	1.1	**Overview** This document outlines essential specifications required for remotely control Apple PPO-Test Auto Tester. A auto tester is a test instrument that applies stimulus on specific areas (x,y) on a DUT (device under test). An automation system could: • Automatically identify and configure the Auto testers • Track the connected testers' status • Load/Unload DUT to/from Auto tester and use the interface for unmanned testing • Apply various testing strategy • Track the DUT testing history
	1.2	**Objectives** This document intends to give clear guidance to supplier on integrating Apple PPO-Test Auto Tester into automation system. • The Supplier is fully responsible for delivering an implementation using the remote control interface to integrate the Auto tester into an automation system, as a requisite to further communication with Apple. • The Supplier is expected to provide design solution with minimum Apple involvement.
	2	**System Structure**
✔	2.1	To start testing, DUTs shall be transmitted from a loading dock to each test station by automated machinery(ie, conveyer belt). When test finishes, DUTs shall be transmitted from each test station to a unloading dock by automated machinery.
✔	2.2	Robot arm or a X-Y-Z-axis scrabbler shall be used for loading/unloading the DUTs to/from Auto tester
✔	2.3	System shall contain several sensors to locate the DUT and read each DUT's SN
✔	2.4	System shall contain an automation server to control and transmit message between different nodes.
✔	2.5	Each testing station shall contain up to 4 tester, and all connected to a Mac-mini as controller. The Mac-mini is connected to Automation server with field control bus. The following contents will focus on defining the interface and protocol between testing station and Automation server.
	2.6	Network When available, we will share our network infrastructure with groundhog, our factory database network.
	2.6.1	Network Diagram with groundhog

		 GroundHog DHCP&DNS Server GroundHog Network Line 1 Automation Server Ethernet(Automation) Robots Sensors Conveyers usb Tester MacMini Ethernet(Groundhog)
	2.6.2	Network Diagram without groundhog and will be used for testing  Line 1 Automation Server Robots Sensors Conveyers Ethernet/CAN usb Tester MacMini

3 Network & Communication

3.1 Network Setup

3.1.1 The network interface between Automation server and testing station should based on Ethernet 100Base-T4, and using TCP/IP protocol(IPv4). The Automation server and testing station are connected to a local network with an Ethernet switch.

3.1.2 IP Address

3.1.2.1 **For Testing:**
The automation server shall use a fixed domain name combined with a fixed IP address. The automation server shall host a DHCP server that assigns IP addresses to the testers' Mac-minis. Use the following address table for reference:

	Domain Name	Netmask	IP	Port
Test Station	N/A	255.255.255.0	DHCP ASSIGNED	Dynamic
Automation Server	Auto.server	255.255.255.0	192.168.1.10	8008

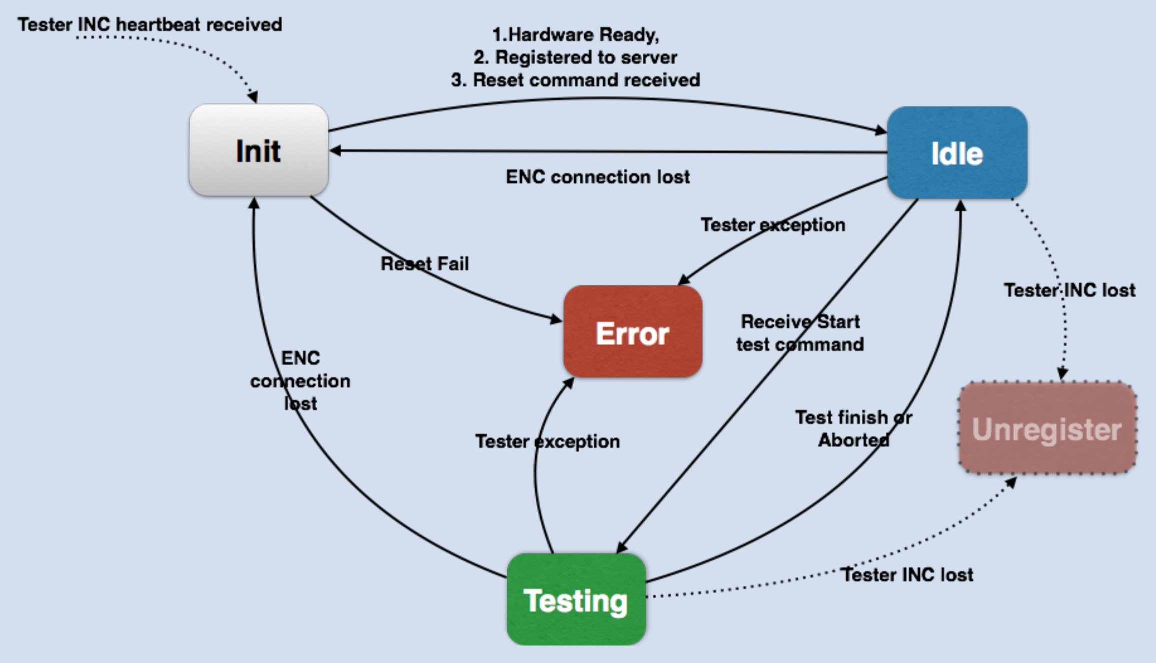
3.1.2.2 **For Engineering build and Mass Production:**
The automation server shall use a domain name and a static IP that's defined with groundhog server admin. The automation server shall host a DHCP server that assigns address to the sensors on the automation network and the DHCP shall operate in a range defined by groundhog admin.

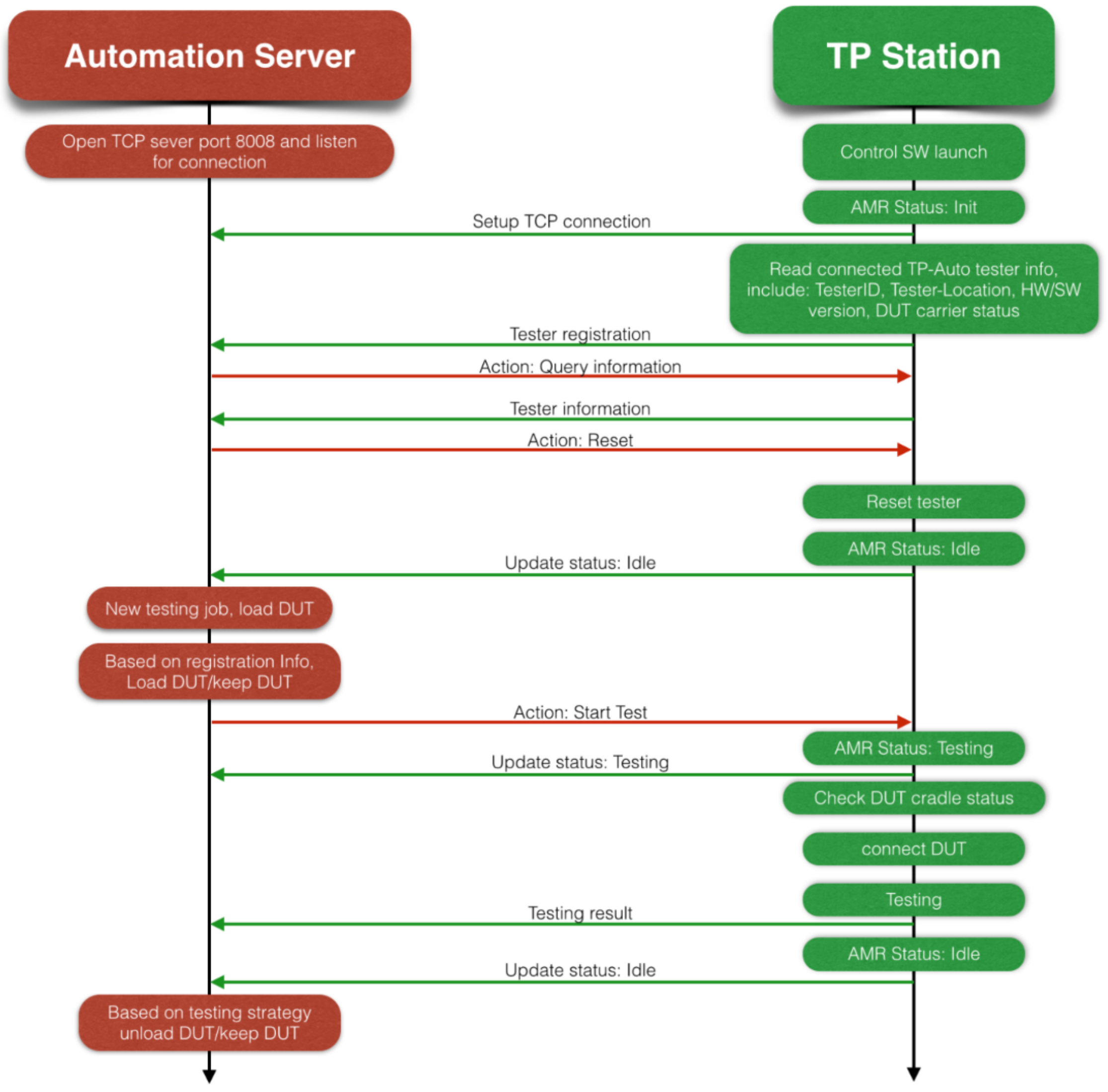
	Domain Name	IP	Port
Automation Server	line_manager	Allocated by groundhog admin	8008
station	N/A	DHCP ASSIGNED by groundhog server	Dynamic

3.2 Test Station Control App

3.2.1 The station contains one Mac-mini and up to 4 Auto testers. An message router like application(AMR) will automatically launch after Mac-mini power-up, it would setup communication using External Network Communication (ENC) with Automation server, and would detect, configure and observe the Auto tester connected to it using Internal Network Communication(INC). It will also keep a state machine to track the status of each tester. The application would also accept command from server, route the message to perform a test, and update the test result back to server after test finish. The following figure describes the state machine of a tester.

3.2.2 Tester State Machine

		 Tester INC heartbeat received 1.Hardware Ready, 2. Registered to server 3. Reset command received Init Idle ENC connection lost Tester exception Reset Fail Error Tester INC lost Receive Start test command ENC connection lost Test finish or Aborted Unregister Tester exception Tester INC lost Testing
	3.3	**Communication and Configuration**
	3.3.1	The Automation server should open a TCP server at port 8008 and listen for test station application to setup TCP connection. The following figure describes the communication procedure.
	3.3.2	Initialization Sequence

Automation Server
TP Station
Open TCP sever port 8008 and listen for connection
Control SW launch
AMR Status: Init
Setup TCP connection
Read connected TP-Auto tester info, include: TesterID, Tester-Location, HW/SW version, DUT carrier status
Tester registration
Action: Query information
Tester information
Action: Reset
Reset tester
AMR Status: Idle
Update status: Idle
New testing job, load DUT
Based on registration Info, Load DUT/keep DUT
Action: Start Test
AMR Status: Testing
Update status: Testing
Check DUT cradle status
connect DUT
Testing
Testing result
AMR Status: Idle
Update status: Idle
Based on testing strategy unload DUT/keep DUT

3.4 Communication Protocol

The communication between Automation server and Test Station is using TCP protocol. Based on that, an application-layer protocol is defined as below for transferring message. The message is transmitted via network using big endian order.

Msg ID(TCP: 32bit / CAN: 29bit)					Msg Data
Reserved	Type	TesterID	SC	SM	Data
6bit/3bit	4bit	20bit	1bit	1bit	1456/8 bytes

3.4.1 Frame description

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	1: Tester Registration
		2: Action
		3: Tester Status
		4: Test Result
		5: Acknowledgement
		6: Heart Beat
		7: Tester information
		0,8-15: Reserved
	Tester ID	000000h~999999h
	SC(only for CAN)	0: Server to Client; 1: Client to Server
	SM(only for CAN)	0: Single packet; 1:Multiple packet
Msg Data	Data length(Byte0&1)	Msg Data total length in bytes
	Data	CAN: Maximum 5bytes/TCP: Maximum 1453bytes
	CRC	CAN: 8bit CRC value/TCP: 0

3.4.2 Tester Registration

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	1: Test Registration
	Tester ID	000000h~999999h
	SC(only for CAN)	1: Client To Server
	SM(only for CAN)	0: Single packet
Msg Data	Byte0-1	Msg Data total length in bytes. Note, length depend on byte2: Register: 664 Unregister: 4

	Byte2	1: Register, 0: Unregister
	Byte3-662	Project/Script[1,20]: Type(1byte):0: None/1: Project/2: Script; Name string(32bytes)
	CRC	CAN: 8bit CRC value/TCP: 0

3.4.3 Action

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	2: Action
	Tester ID	000000h~999999h
	SC(only for CAN)	0: Server to Client
	SM(only for CAN)	1: Multiple packet(Start test) 0: Single packet(Other)
Msg Data	Byte0-1	Msg Data total length in bytes
	Byte2	1: Start test 2: Stop test 3: Reset 4: Query tester status 5: Query tester information
	Byte3~34	Optional for start test: Project name in ASCII c string (32 Bytes) Leave all 0s if not used. File extension is not included in this field.
	Byte35~66	Optional for start test: Script name in ASCII c string(32 Bytes) Leave all 0s if not used. File extension is not included in this field.
	Byte67~322	Optional for start test: SerialNumber name in ASCII c string(256 bytes) Leave all 0s if not used. Additional information is allowed in SerialNumber field, use “;” for separation.
	CRC	CAN: 8bit CRC value/TCP: 0

3.4.4 Tester Status

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	3: Tester Status
	Tester ID	000000h~999999h
	SC(only for CAN)	1: Client To Server
	SM(only for CAN)	0: Single packet
Msg Data	Byte0-1	Msg Data total length in bytes, 4
	Byte2	0: Init 1: Idle 2: Testing 3: Error

			CRC	CAN: 8bit CRC value/TCP: 0

3.4.5 Test Result

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	4: Test Result
	Tester ID	000000h~999999h
	SC(only for CAN)	1: Client To Server
	SM(only for CAN)	0: Single packet
Msg Data	Byte0-1	Msg Data total length in bytes
	Byte2	0: Fail 1: Pass
	Byte3-6	Error Code
	Byte7~38	Error String(32bytes)
	Byte39~294	256 Bytes DUT SN
	CRC	CAN: 8bit CRC value/TCP: 0

3.4.6 Acknowledgement(Only for CAN)

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	5: Acknowledgement
	Tester ID	000000h~999999h
	SC(only for CAN)	0/1: Both
	SM(only for CAN)	0: Single packet
Msg Data	Byte0-1	Msg Data total length in bytes
	Byte2	0: NACK 1: ACK
	Byte3~	NACK reason(TBD)
	CRC	CAN: 8bit CRC value/TCP: 0

3.4.7 Heart Beat

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	6: Heart beat
	Tester ID	0: Server, Non-0: tester
	SC(only for CAN)	0/1: Both
	SM(only for	1: Multiple packet

	CAN)	
Msg Data	Byte0-1	Msg Data total length in bytes: 17
	Byte2-15	TimeStamp(YYYYMMddHHmmss)
	CRC	CAN: 8bit CRC value/TCP: 0

3.4.8 Tester Information

Field		Description
Msg ID	Reserved	CAN:0b000/TCP:0b000000
	Type	7: Tester information
	Tester ID	000000h~999999h
	SC(only for CAN)	1: Client to Server
	SM(only for CAN)	1: Multiple packet
Msg Data	Byte0-1	Msg Data total length in bytes
	Byte2	Tester Location (1,255)
	Byte3	0: Not occupied, 1: Occupied
	Byte4~	various HW/SW version name and version value combination, each combination is separated by space and version name <key_word> and version value <version_string> is separated by ':' (e.g.: Tester_HW:111111 Tester_FW:222222 GaiaVersion:GAIA-53.2.0 GRPVersion:1.0.46 TestVersion:0.2.14 UIExploreVersion:0.0.1 GTSVersion:0.01 AMRVersion:0.0.1)
	CRC	CAN: 8bit CRC value/TCP: 0

4 Test Management

4.1 Summary

The Automation server should maintain a database on a Mac-system, track all the testing data for further maintenance and trouble shooting. The database entry should be based on a single testing.

4.2 Yield and Retest Management

✔	4.2.1	The server shall compute per tester yield information, provide statistic report based on testing error code
✔	4.2.3	The server shall store per unit testing history
✔	4.2.3	The server shall store per carrier testing history

✔	4.2.4	Data Base Entries(Detailed list will be given in a separate document, below is just an example)

Columns	Data Type(size)	Description	Default	Reserved Value(description)	NOTE
Test Result Entry Number(unique)	Unsigned Integer(8 bytes)	Auto increment value for each data entry	0		
BuildName	Text(32 char)	Build names like PVT, EVT and etc appended with build count	NULL	'DROP#' 'PVT#' 'EVT#' 'DVT#' 'PVT#' 'RAMP#' 'MP'	# in Reserved values represent an integer value
GAIA Version	Text(128 char)	GAIA version	NULL		
GRPTest version	Text(128 char)	GRPTest version	NULL		
G2 version	Text(128 char)	G2 version	NULL		
UIExplore Version	Text(128 char)	UIExplore Version	NULL		
AMR Version	Text(128 char)	AMR Version	NULL		
ServerVersion	Text(128 char)	ServerSoftware version	NULL		
SerialNumber	Text(256 char)	DUT serial number	NULL		
TesterID	Unsigned Integer(4 bytes)	TesterID	NULL		
JavisID	Unsigned Integer(4 bytes)	Javis ID	NULL		
ServerID	Unsigned Integer(4 bytes)	ServerID	NULL		
ServerIP	Text(256 char)	ServerIP	NULL		
StartTime(unique)	DateTime	Time when the server issues the start test	NULL		time on server
StopTime	DateTime	Time when the tester responds with the result	NULL		time on server
Error Code	Unsigned Integer(2 bytes)	Test result error code	NULL	0 (No error)	
Error String	Text(256 char)	Test result error string	NULL	'PASS'(pass)	
LogFile Name	Text(4096 char)	Log and plist file name(full path)	NULL		

		<table><tr><td>**ParameterName0 ParameterName0_ul ParameterName0_ll ... ParameterNameX ParameterNameX_ul ParameterNameX_ll**</td><td>Double</td><td>Parameter names, upper limits, and lower limits</td><td>NULL</td><td></td><td>This could be put in a different table associated with 'Test Result Entry Number'</td></tr></table>
	4.3	**Tester Health Monitoring**
✔	4.3.1	Based on the tester yield information, the server should acknowledge the tester' healthy status and has the ability put a particular tester to un-healthy state and stop using it under particular situation. A definition of the situation would be given later.
✔	4.3.2	The automation server shall take a configuration file that specifies a group of error codes that indicate tester issues.
	4.4	**Retest Strategy**
	4.4.1	**Retest terminology.** The sequence of retest is specified with a sequence of case insensitive letters ranging from 'a-z'. The first letter of the retest represents the first tester in which the DUT failed on. Same letter in the sequence represents the same tester. For example, 'aab' means if DUT should fail on tester 'a', it shall be retested on the same tester 'a' again. If it fails again, then the DUT shall be tested on a tester different than tester 'a'.
✔	4.4.2	The automation server shall take a configuration file that configures the retest rules per different error codes.
✔	4.4.3	For error codes not specified in the configuration file mentioned in 4.7.1, the default retest strategy shall be 'aab'.
✔	4.4.4	The server software shall notify the user for some human typo errors, which include but not limited to: • Same error code has two different retest strategy • Error code exceeds 4 digits
✔	4.4.5	The server shall be able to pick at least 3 other testers for every tester in the system. The server shall follow the normal tester choosing strategy specified in section 4.5 when picking a different retest tester.
	4.5	**Tester Choosing Strategy**
	4.5.1	**Normal Testing Mode** This is the mode we run during normal testing operations
✔	4.5.1.1	When to perform a testing, the Automation server should always choose a tester by following manners: 1. The tester is idle at the moment. 2. The chosen tester doesn't need health check 3. The chosen tester shall have the least executed the fewest tests
✔	4.5.1.2	When an audit DUT is being tested, it waits on the tester until the next designated tester is idle. The robot then picks up the audit DUT and places it on the direct next tester.
Need to finalize POR cleaning	4.5.1.4	When a stimpad-cleaning module is being tested, it waits on the tester until the next tester is idle. The robot then picks up the cleaning module and places it on the direct next tester.

strategy		
	4.5.2	**GRR Mode** This is the mode when we need to run multiple DUTs on the selected testers multiple times.
✔	4.5.2.1	The server shall take rotate the DUTs on each tester until all DUTs are tested the configured number of times. Typically we require 5 DUTs to run 5 times on each tester. For example, if we have 3 DUTs, and we need to run each DUT 2 times on each tester. For each tester, we run unit x->y->z->x->y->z
✔	4.5.2.1	The server shall take a configuration file that specifies which testers need to be GRR'ed. The default is all testers.
	4.5.3	**Manual Mode** This is the mode when we want to manually select a tester to test DUT
✔	4.5.3.1	Manually let user pick a tester for each DUT
	4.5.4	**Test and Project Script Selection**
✔	4.5.4.1	Automation server shall let user pick which test and project to use when executing test. User shall be able to specify the configuration that applies to all testers not configured.
✔	4.5.4.2	Apple will specify the script to run for stimpad cleaning, daily audit, and GRR at the beginning of each build.
	4.6	**Testing Log&Data Management**
Not needed for initial deployment	4.6.1	The automation server shall provide an interface for tester station uploading testing results including result file(csv) and log. And the whole automation system should be able to be deployed on an Apple Groundhog system and the collected testing result could be uploaded to Apple PDCA system.
	4.7	**Auto Tester Maintenance**
✔	4.7.1	The server shall be able to be configured to clean tester stimpads periodically.
✔	4.7.2	Cleaning material and methodology shall be approved by Apple. Apple will provide a cleaning plan by default.
	4.8	**DUT Binning**
✔	4.8.1	Before MP, the system shall support a minimum of 5 bins.
✔	4.8.2	The server shall take a configuration file that maps different error code to different bins. Default binning for not-configured error codes shall be pass and fail.
	4.9	**Critical Hardware/Software Failure Management**
✔	4.9.1	The server shall keep the status of all DUTs inside the system on the hard drive. In case of a critical error or crash, the locations of all DUTs can be obtained.

✔	4.9.2	Upon the recovery of crash, all passed DUTs shall be sent to pass bin. All other DUTs shall be sent to fail bin for the operator to re-input into the system.
	5	**Server Setup and Maintenance**
✔	5.1	The server software including the database shall be configured to work on Mac OSX 10.10 or later
Not needed for initial deployment	5.2	All the necessary software shall be packaged in one package with a one page Apple approved installation procedure
✔	5.3	All the configuration parameters used by server software shall be maintained in a single plist file.
✔	5.4	The server shall connect or disconnect to any number of testers during any time of the operation. In other words, testers is hot pluggable to the network without interfering with other parts of the automation.
	6	**Server Operator GUI Features**
✔	6.1	**Tester online/offline indicator** The server GUI shall have an indicator showing the operator when a tester location has a tester online or offline.
✔	6.2	**Tester maintenance indicator** The server GUI shall have an indicator showing the operator when a tester needs to be serviced. Generally, the server will know this by different error codes sent back by the tester.
✔	6.3	**Tester state indicator** The server GUI shall have an indicator showing the operator different states of the connected testers. The states shall include by not limited to the following list: • Idle • DUT loading • DUT loaded • Testing • DUT unloading • Error
✔	6.4	**List of sw versions for testers online** When queried by the operator. The server shall present a list including the following information for each tester: • SW version • FW version • PLC FW version
✔	6.5	**List of sw&hw versions for automation system** When queried by the operator. The server shall present a list including the following information for the automation system: • All PLC FW versions • Server SW version

		• Database SW version

7 Update History

Date	Section	Change	Modified By
05/05/2015	4.1, 4.5, 4.6, 5, 6	Added more details to test management. Added GUI requirements. Added more details to section 5.	SY
05/28/2015	3.2, 3.4	Updated state machine Updated frame description Fix typo	QZ
05/28/2015	4.2.4, 4.9	Minor format cleanup. Updated critical failure management	SY
06/08/2015	3.4.3, 3.4.6, 3.4.9	ENC protocol update	QZ
06/09/2015	3.4.1	ENC protocol update. Typo fixes at a few sections.	SY
06/22/2015	3.4, 3.4.5, 4.4	Add endianess requirement to network message. Update ENC format in 3.4.6 Fix numbering typo under 4.4.	QZ
07/06/2016	3.4.3, 3.4.8	Update version string specification Update file extension comments in project name and test script name field Add additional info specification in SerialNumber field	NZ
10/02/2017	3.4.3, 3.4.5	Update SN length to 256 bytes	QZ