CSIR - Distributed Application Manager
Functional Requirements and Design Document

101 Solutions

September 13, 2013

Version 0.9

Francois Germishuizen	11093618
Jaco Swanepoel	11016354
Henko van Koesveld	11009315

Change Log

Date	Version	Description
13 Sept	Version 0.1	Document Created
13 Sept	Version 0.2	Added usecase diagrams
13 Sept	Version 0.3	Added AddBuild, RequestSysInfo and GetSysInfo usecase
13 Sept	Version 0.4	Added Upcoming usecases and Simulations
13 Sept	Version 0.5	Added overall processes and adjusted margins
13 Sept	Version 0.6	Added AddSlave, StartServer, StopServer and SetPort usecase
13 Sept	Version 0.7	Added Glossary
13 Sept	Version 0.8	Added Class diagrams
13 Sept	Version 0.9	Added AddBuild via network usecase

Contents

1 Overview

1.1 Background

The CSIR is actively developing a distributed simulation framework that ties in with various other real systems and is used to exchange information between them. The client has a number of configurations of this system depending on the requirements of the client which can involve various external applications as well.

One of the issues the client has is to quickly distribute the latest build or configuration files of their software over various computers that are needed for an experiment. In some cases the same computers may be used for other experiments which mean each of the computers may need to have various builds and configuration options.

Another issue they experience is the running, stopping and restarting of the complete simulation. During a simulation it may be determined that certain configuration options may need to be changed and distributed to the affected machines, in which case either all or some components will need to be restarted which can become tedious and time consuming.

1.2 Business opportunity

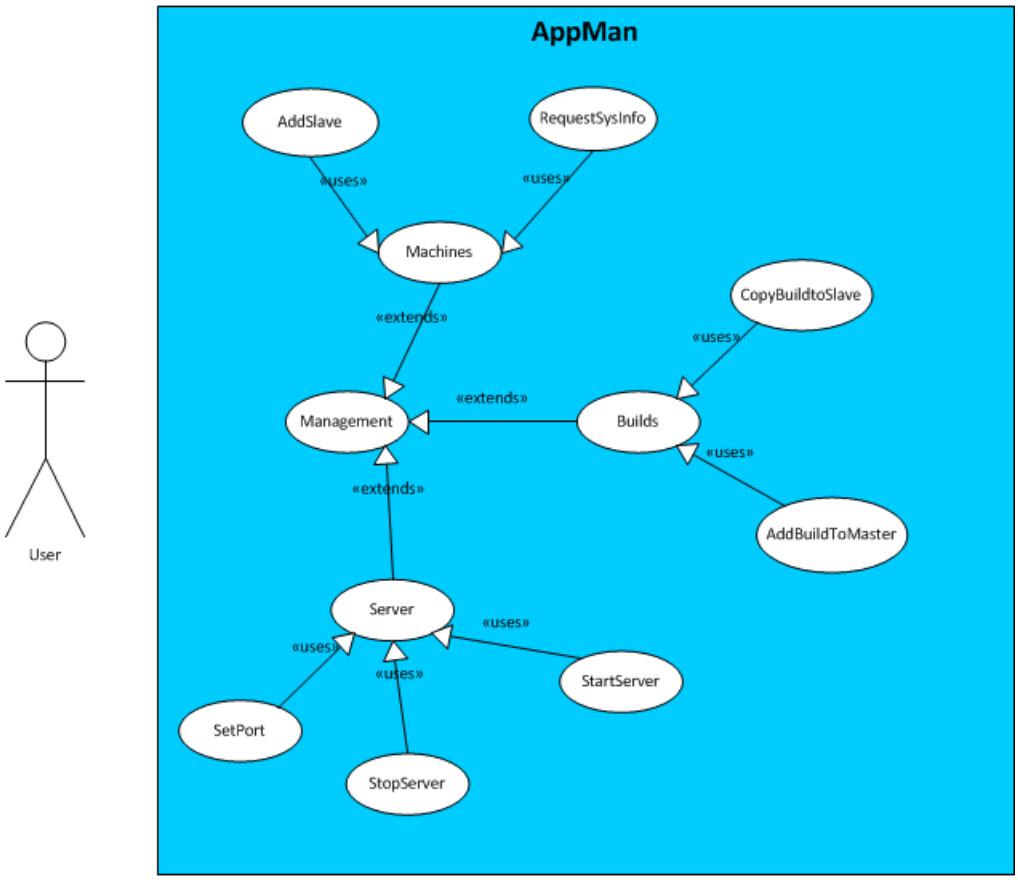
The goal of our project is to develop an application which is able to maintain various build versions of the simulation framework and distribute these builds to certain designated machines that may be required for an experiment. The application will monitor system statistics of the various machines attached to an experiment and will have the ability to execute applications on those machines which will have different configuration options.

The application will consist of a master and slave component where the master is used to control the distribution of slaves. From the master one will be able to start an experiment which will run the relevant applications on all the necessary machines.

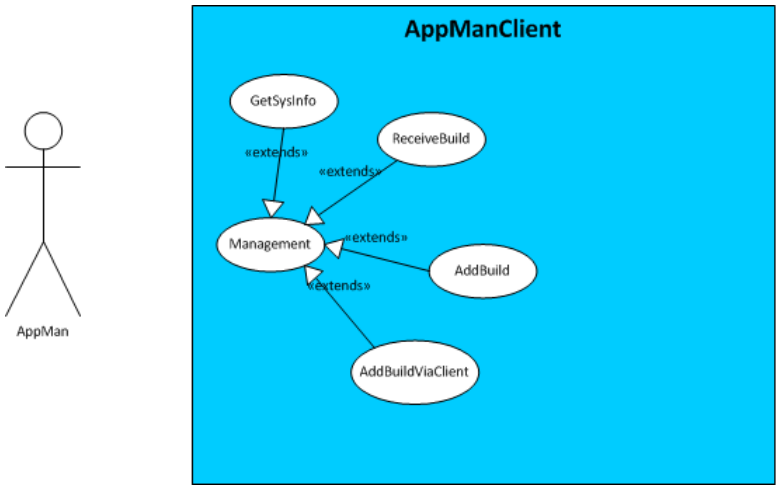
2 Use Cases

We have both a master and slave application. The master is called AppMan and the slave is called AppManClient.

Below is a use case of the AppMan system as it currently is:

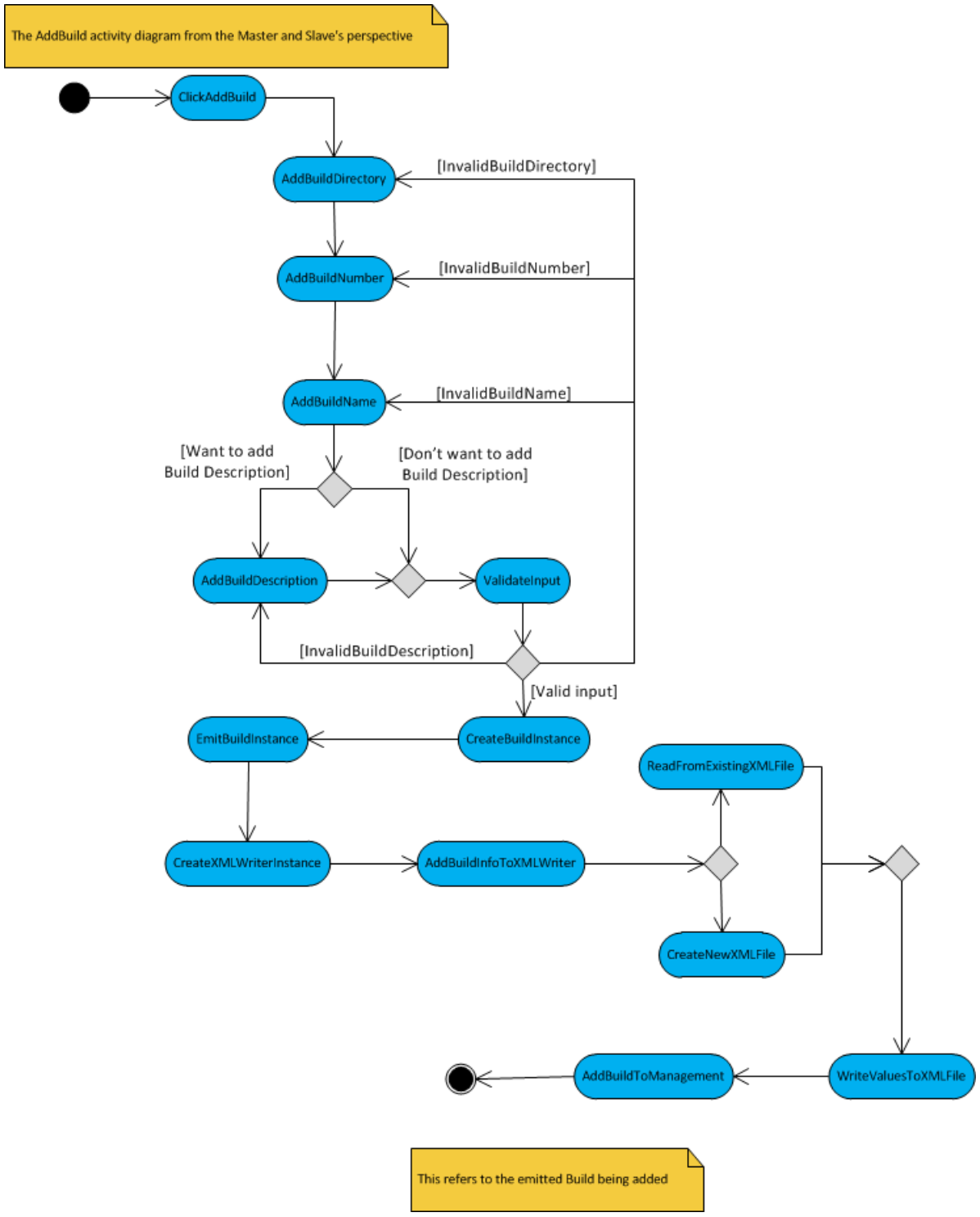


Below is a use case diagram of the AppManClient system as it currently is:



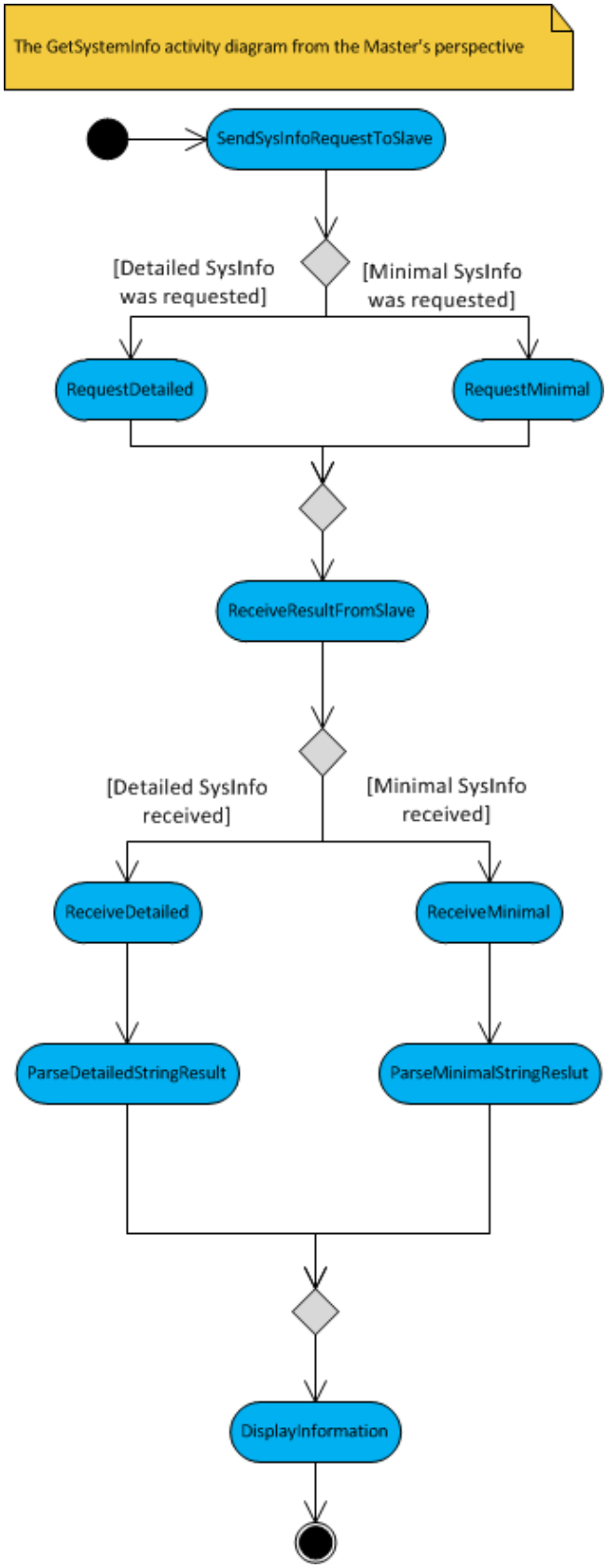
2.1 AddBuild usecase

The following activity diagram is applicable to both AddBuildToMaster in AppMan as well as AddBuildViaClient in AppManClient.



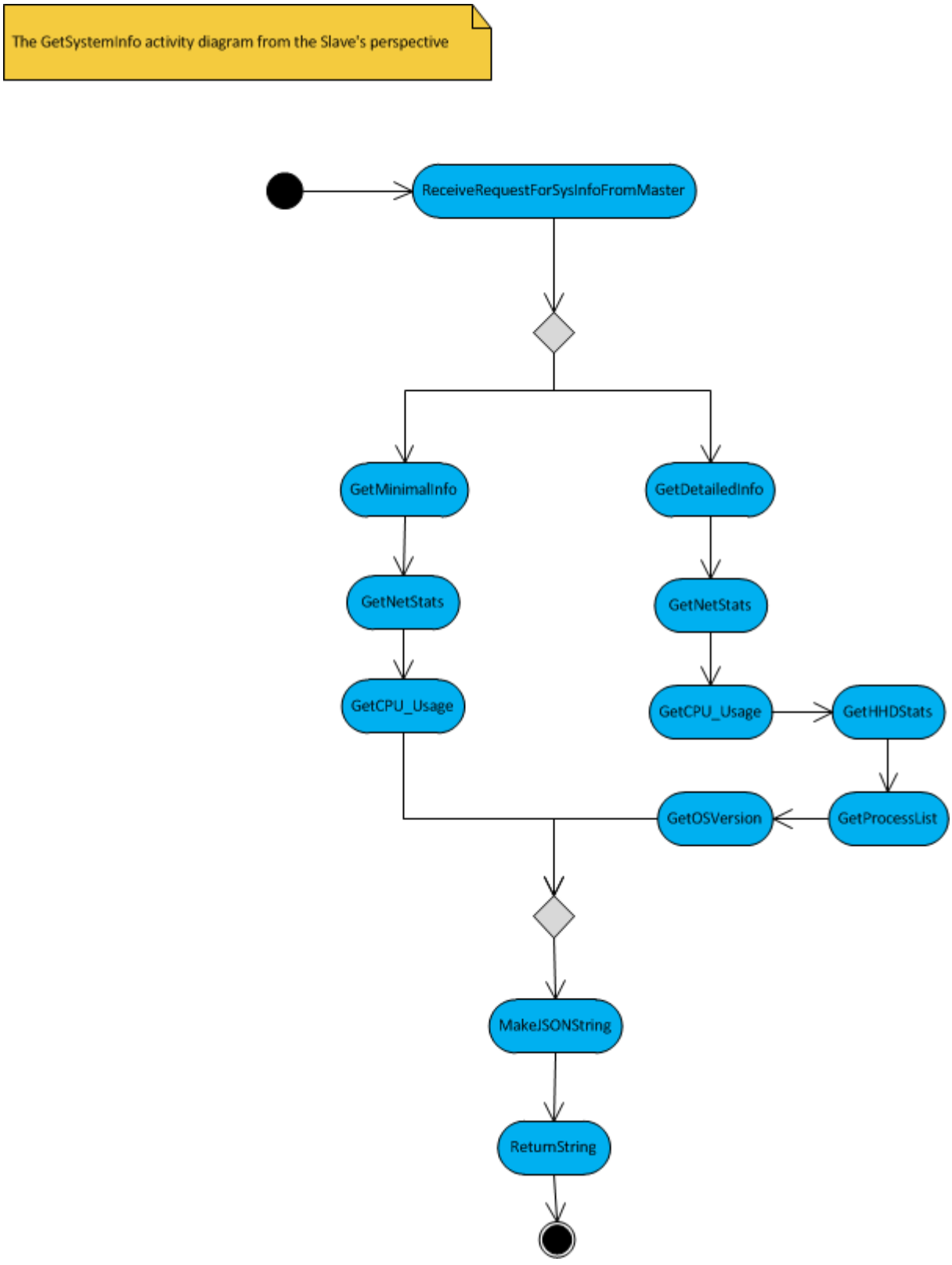
2.2 RequestSysInfo usecase

The following activity diagram is applicable to the RequestSysInfo usecase in AppMan.



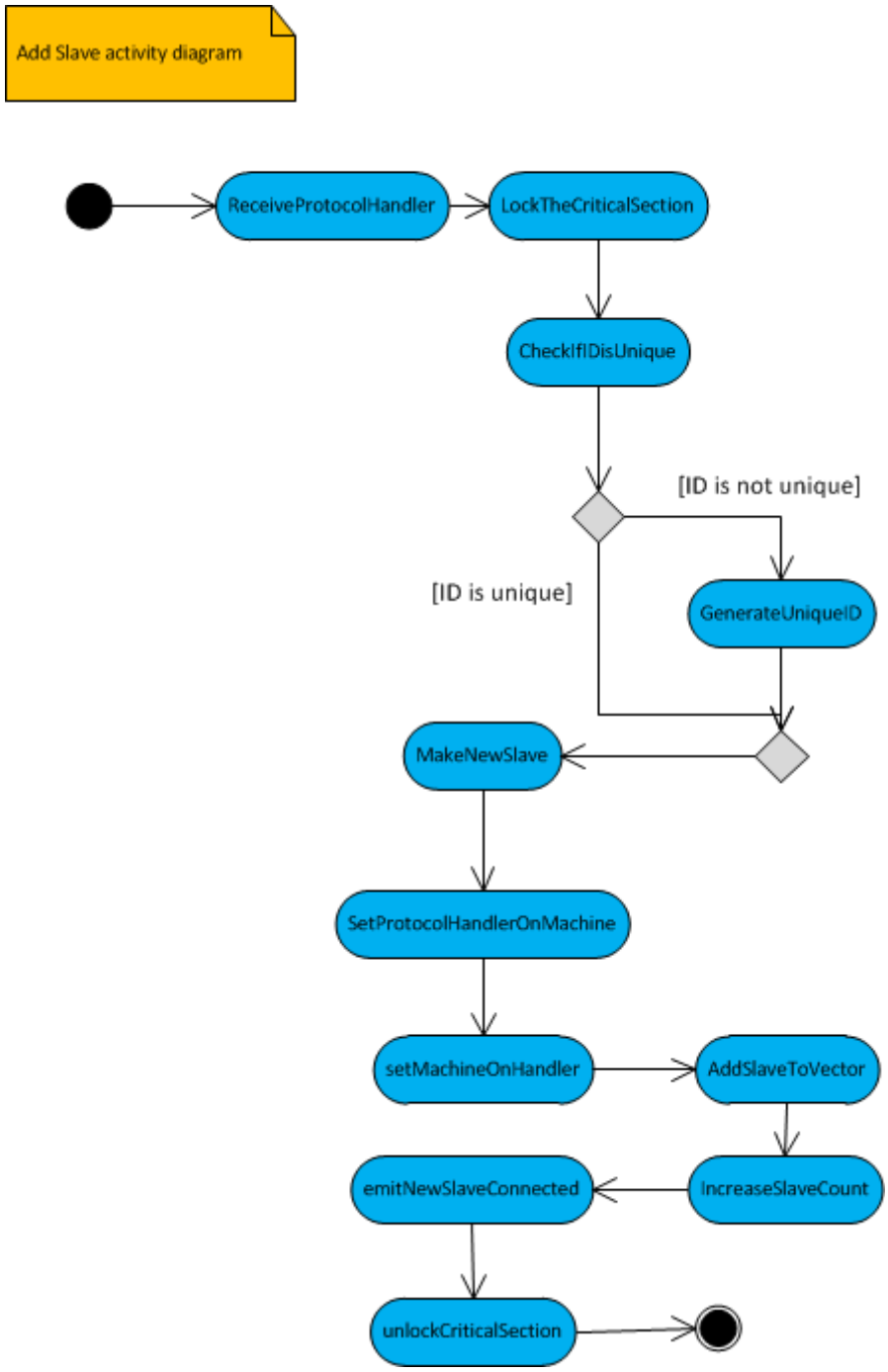
2.3 GetSysInfo usecase

The following activity diagram is applicable to the GetSysInfo usecase in AppManClient.



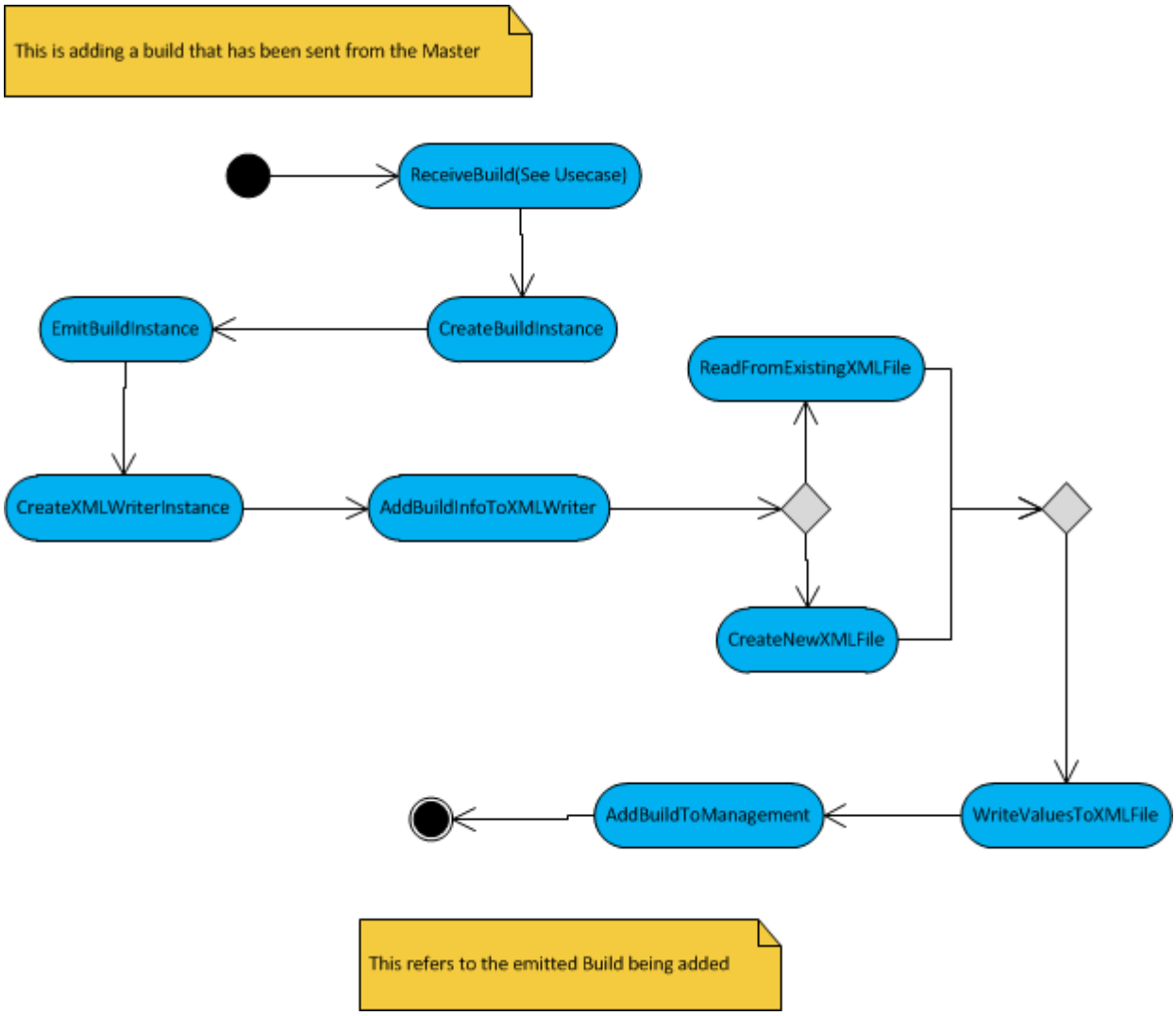
2.4 AddSlave usecase

The following activity diagram is applicable to the AddSlave usecase in AppMan.



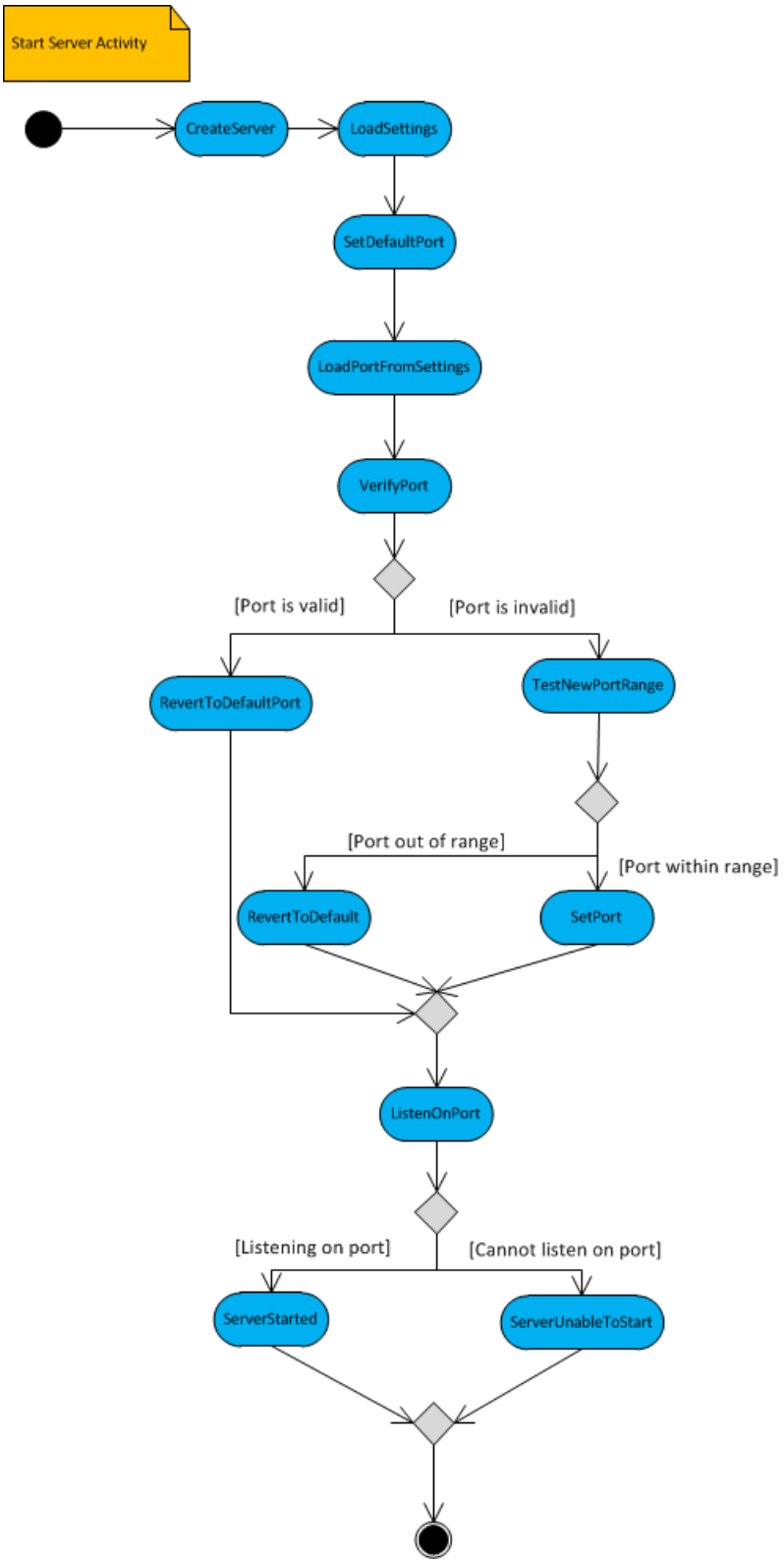
2.5 AddBuild via network usecase

The following activity diagram is applicable to the AddBuild usecase in AppManClient.



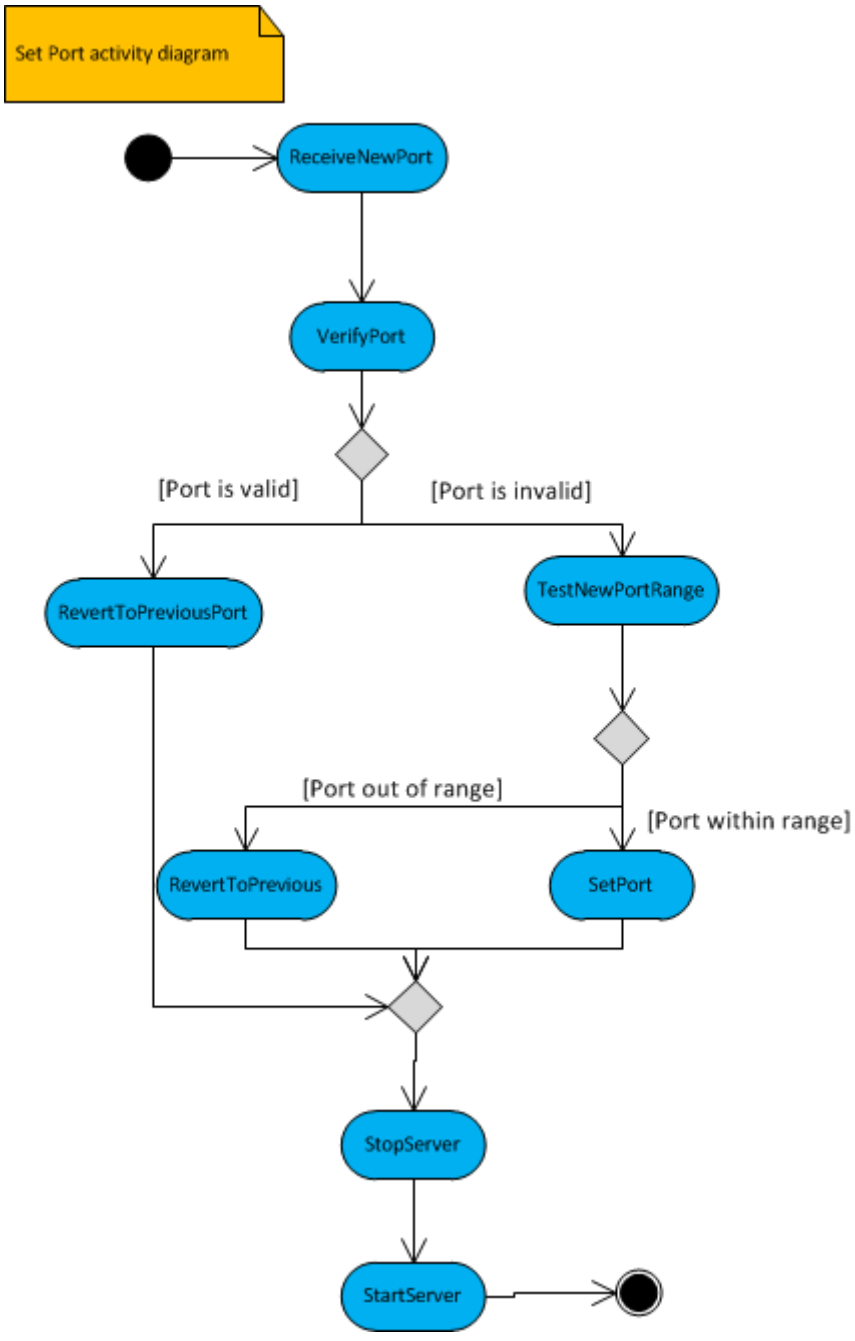
2.6 StartServer usecase

The following activity diagram is applicable to the StartServer usecase in AppMan.



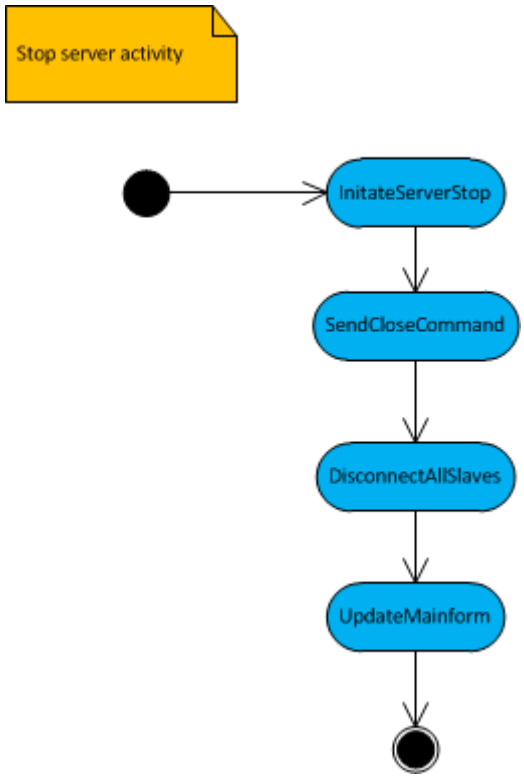
2.7 SetPort usecase

The following activity diagram is applicable to the SetPort usecase in AppMan.



2.8 StopServer usecase

The following activity diagram is applicable to the StopServer usecase in AppMan.



3 Upcoming usecases

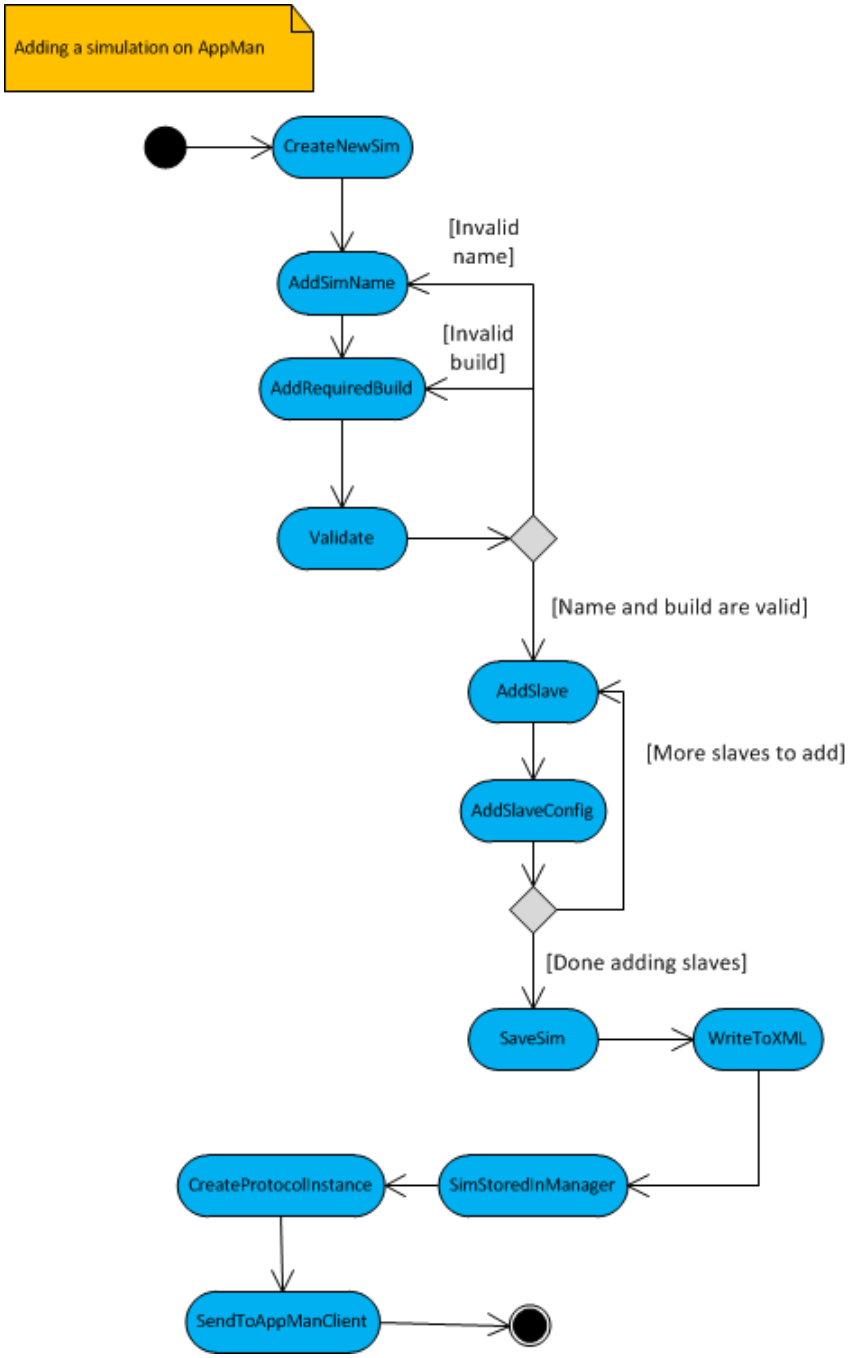
This section is for the documentation of soon to be implemented usecases.

3.1 Simulations

Simulations are a set of builds to be run in specific order on a set of slaves.

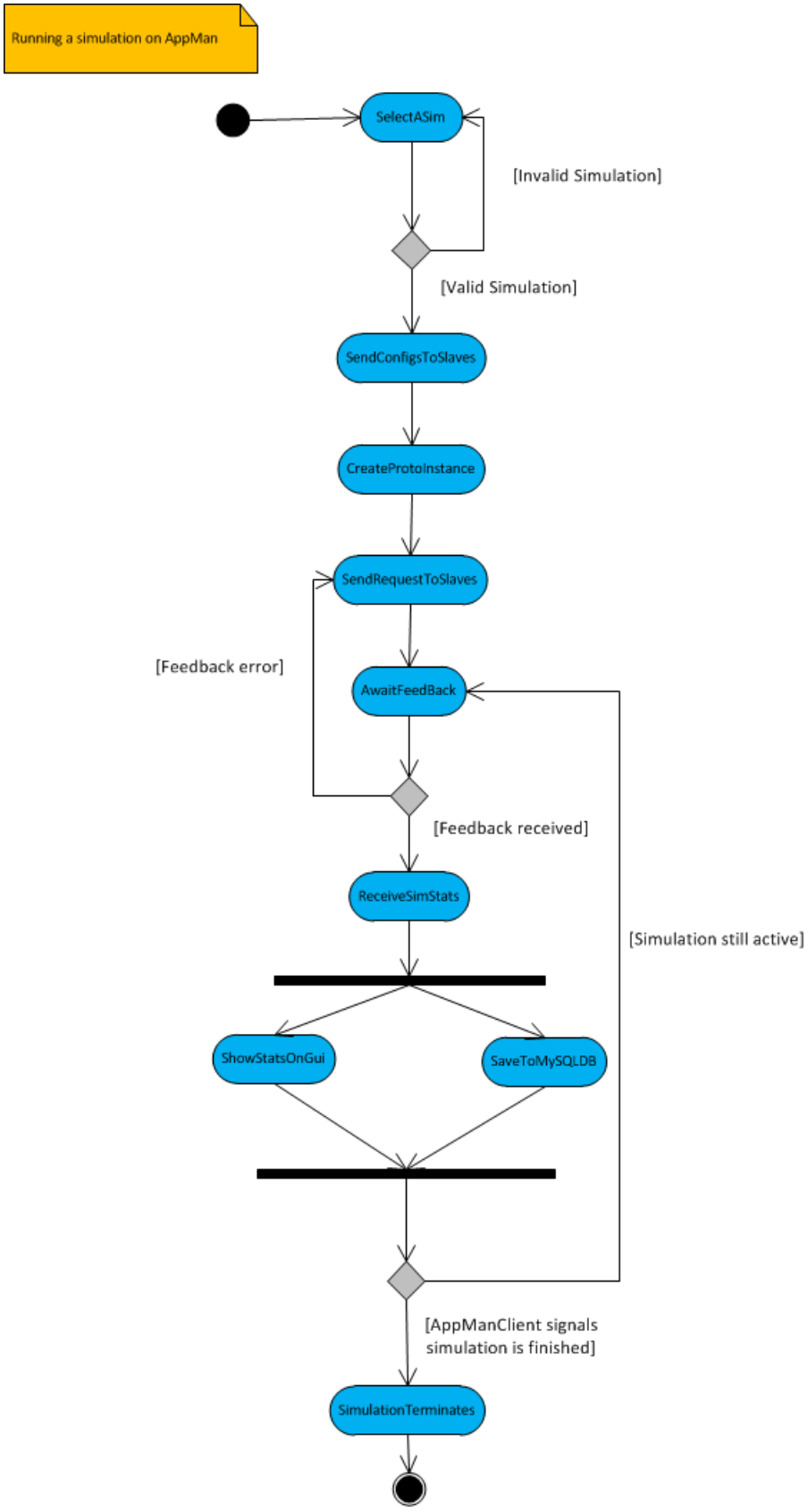
3.1.1 AddSimulation

A simulation must be added to AppMan in the following way:



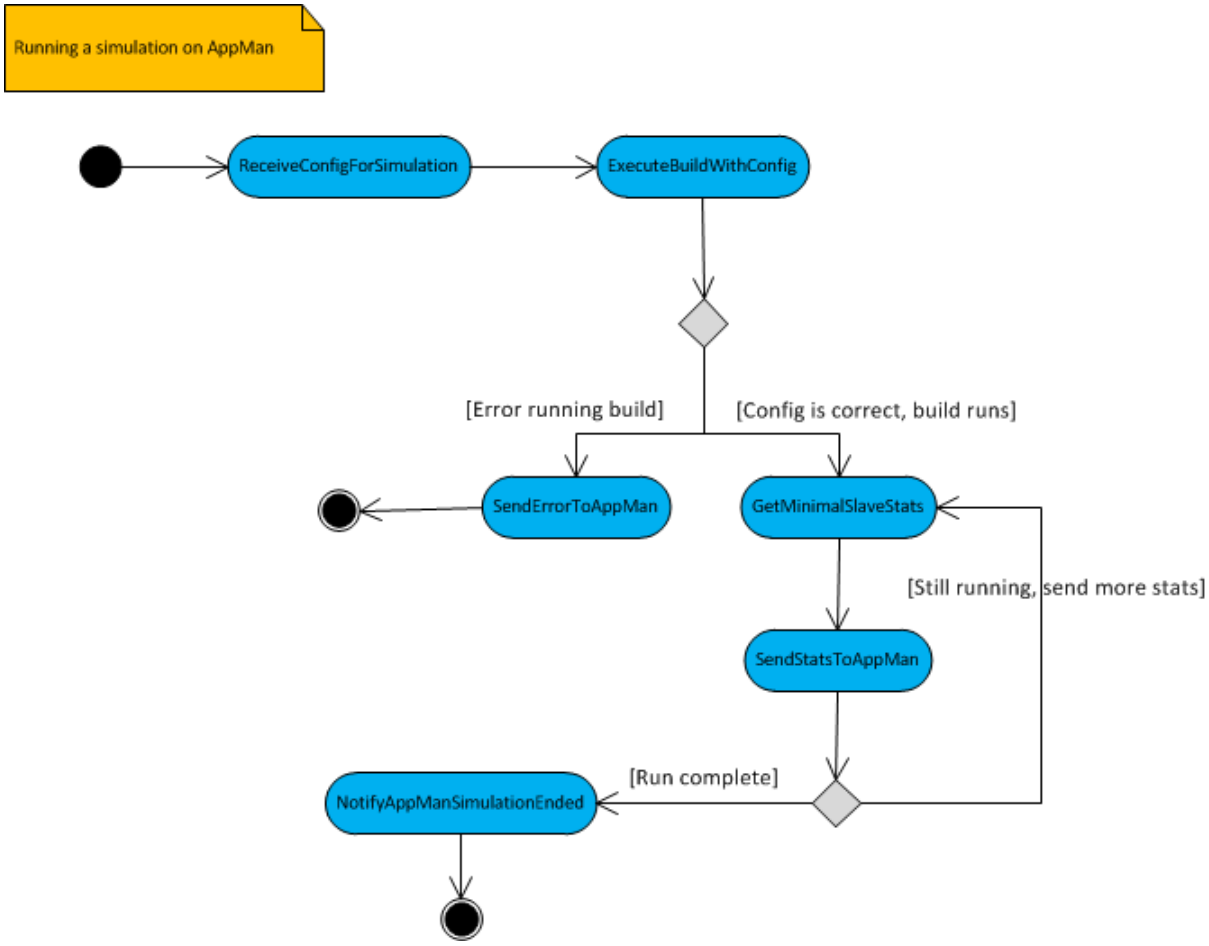
3.1.2 RunSimulation AppMan

A simulation is run from AppMan in the following way:



3.1.3 RunSimulation AppManClient

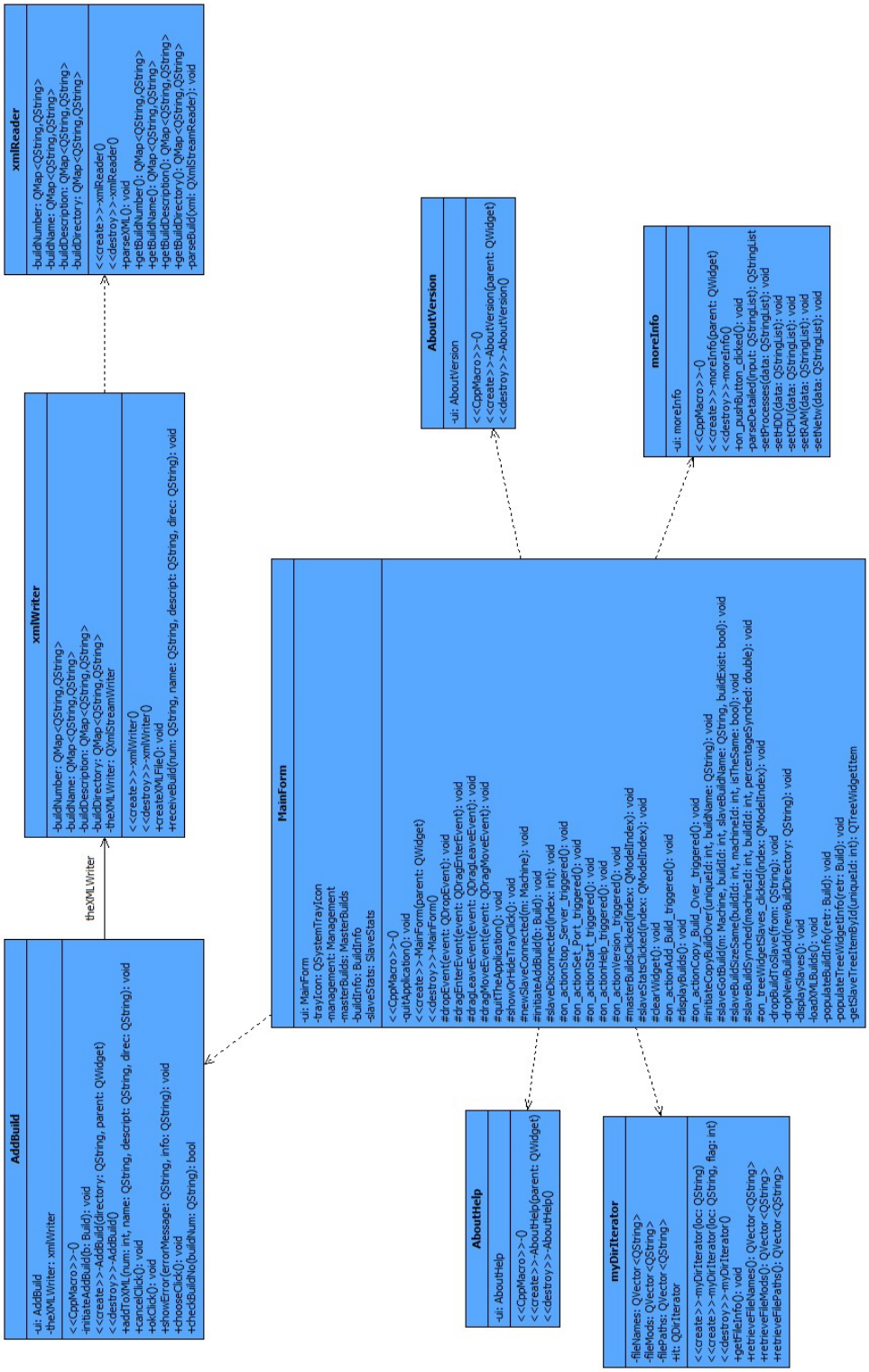
A simulation is run from AppManClient in the following way:

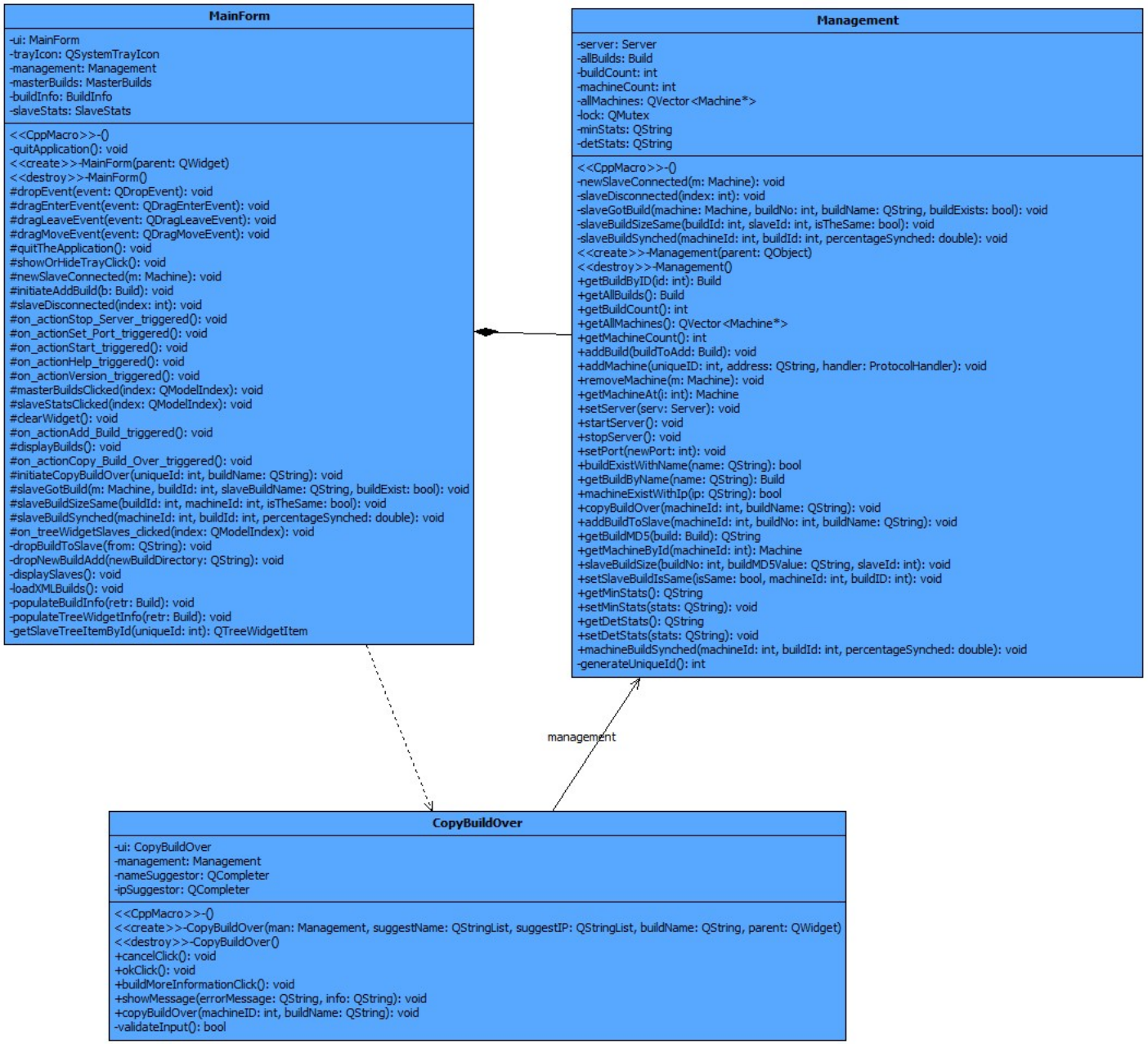


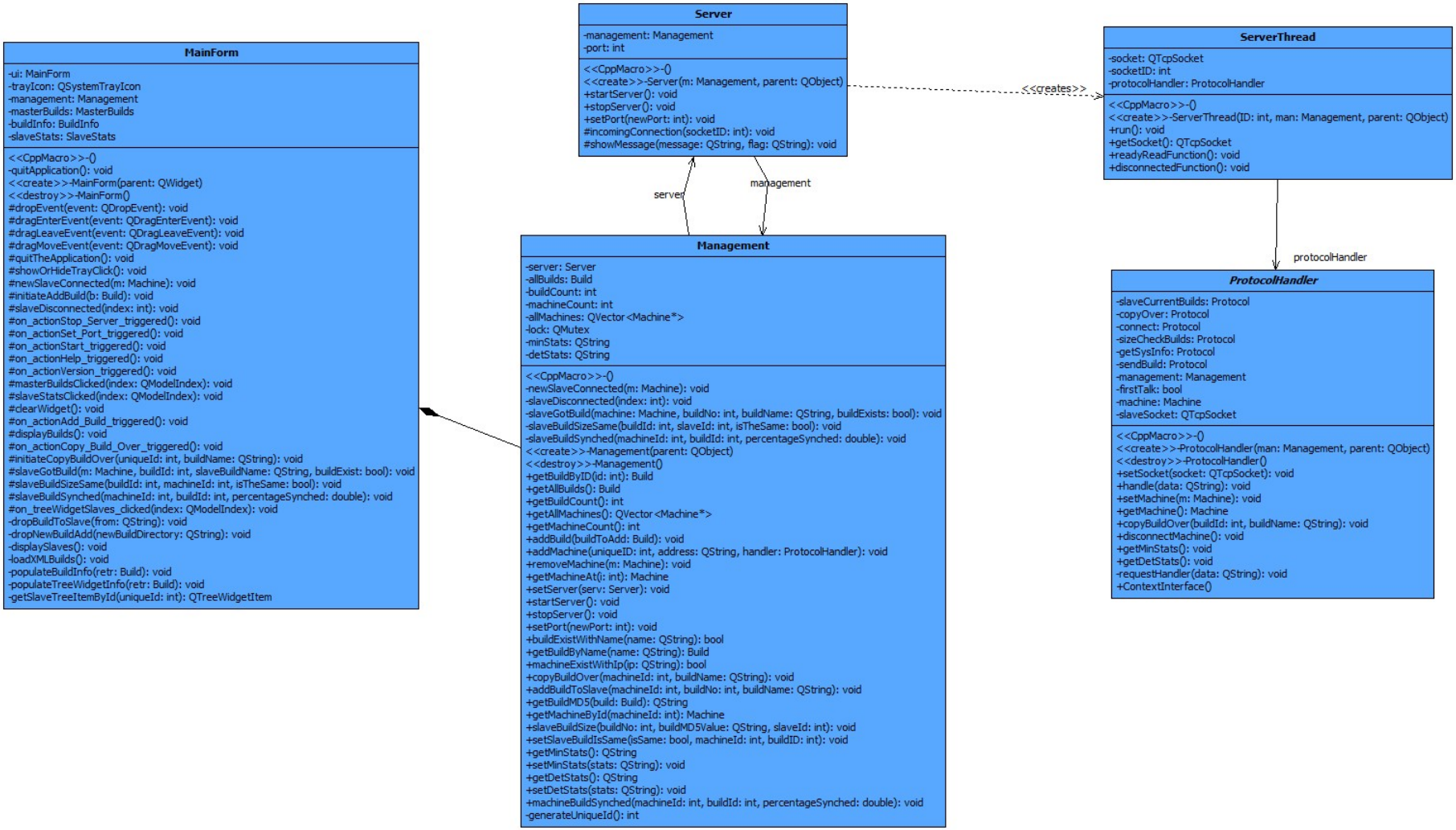
4 Class diagrams

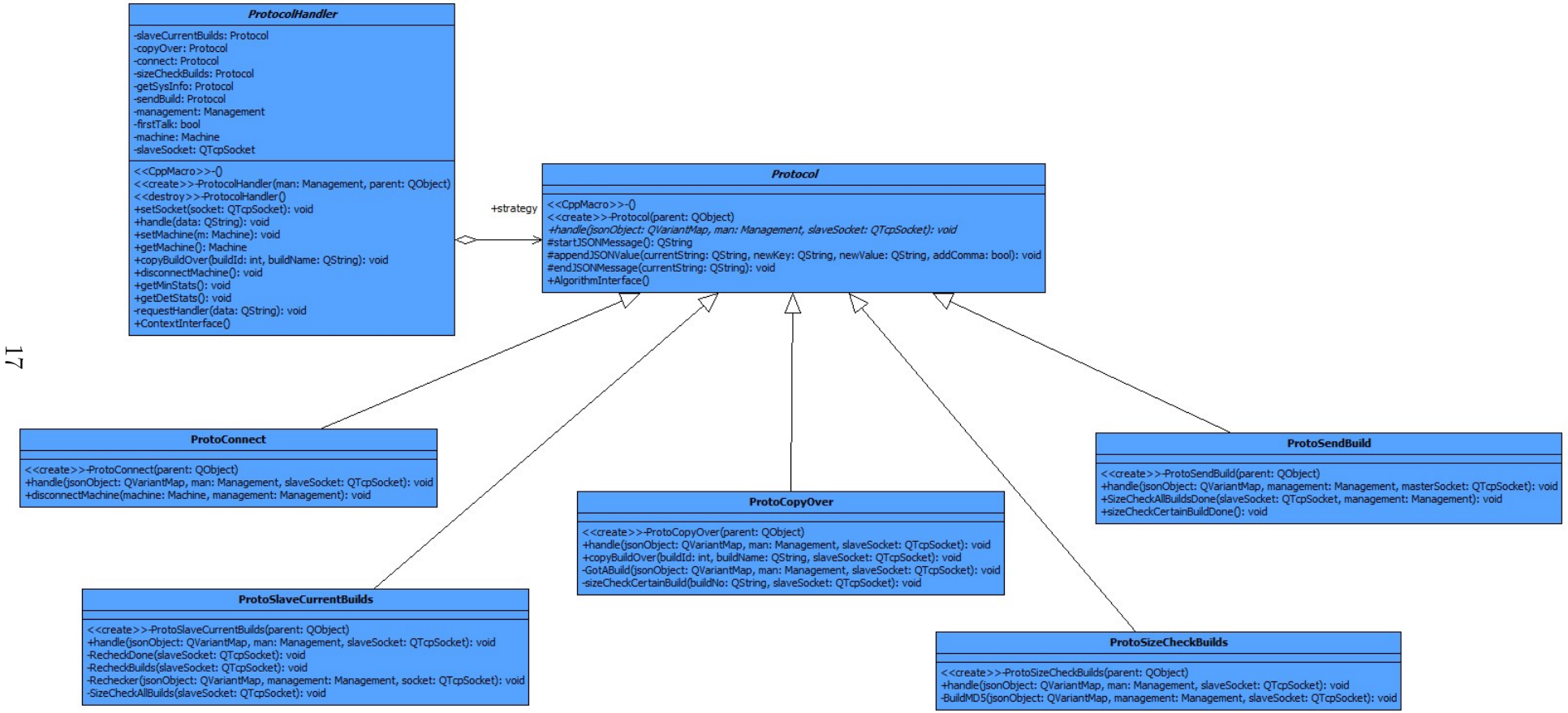
4.1 AppMan Class Diagram

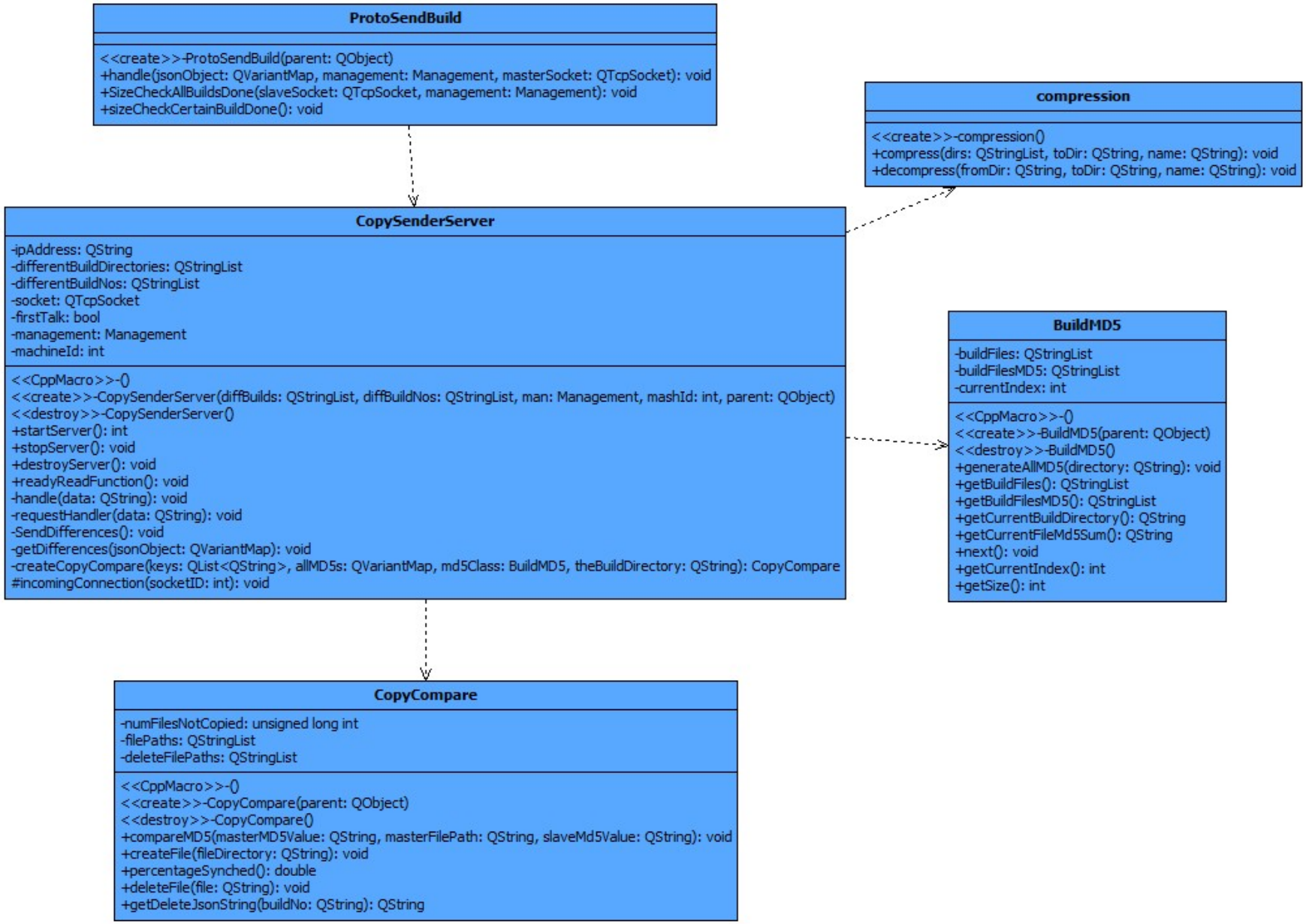
Due to its large size, the class diagram is split into multiple, seperate diagrams. Classes will be repeated to show how they all link.

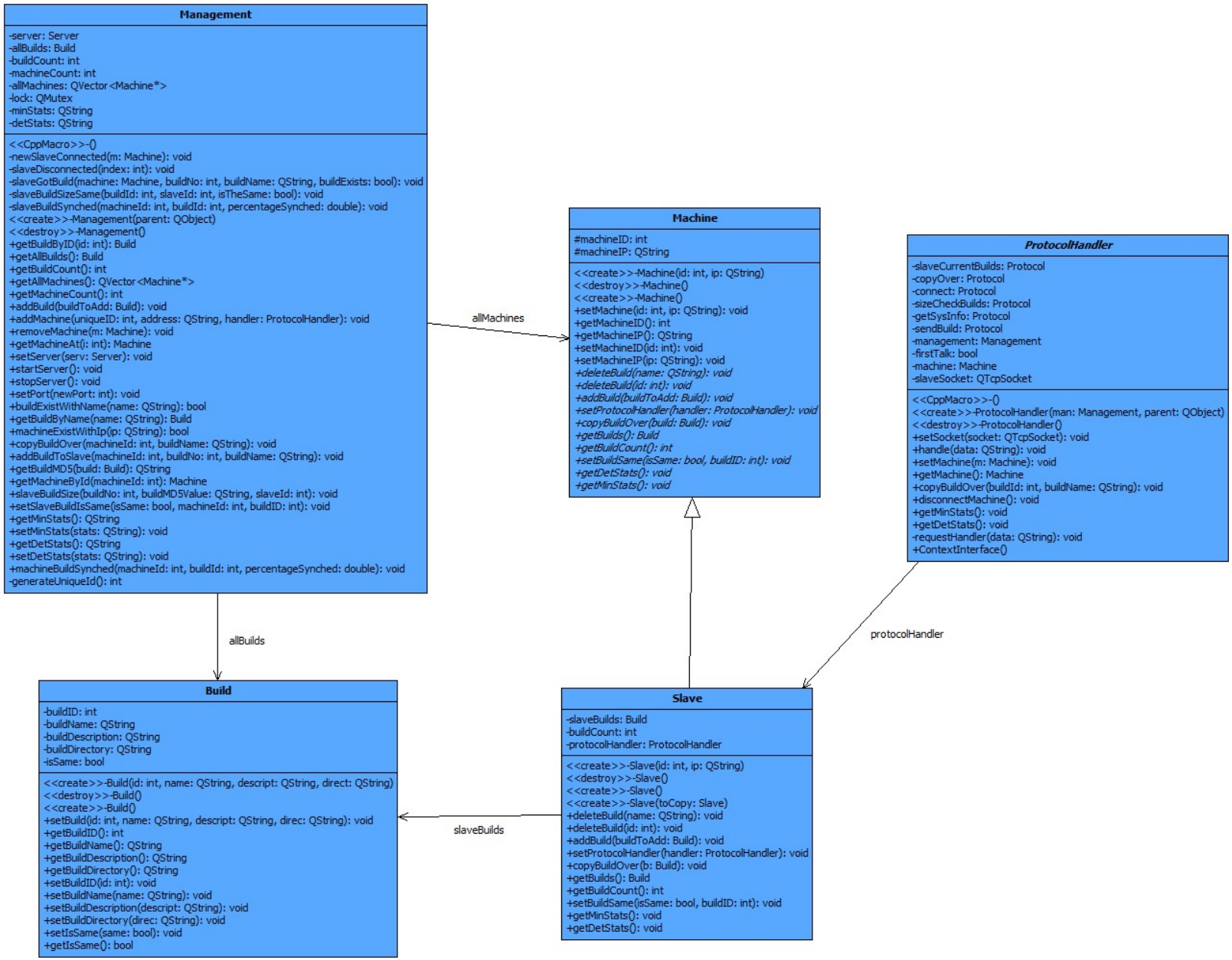



MainForm
-ui: MainForm
-trayIcon: QSystemTrayIcon
-management: Management
-masterBuilds: MasterBuilds
-buildInfo: BuildInfo
-slaveStats: SlaveStats
<<CppMacro>>-()
-quitApplication(): void
<<create>>-MainForm(parent: QWidget)
<<destroy>>-MainForm()
#dropEvent(event: QDropEvent): void
#dragEnterEvent(event: QDragEnterEvent): void
#dragLeaveEvent(event: QDragLeaveEvent): void
#dragMoveEvent(event: QDragMoveEvent): void
#quitTheApplication(): void
#showOrHideTrayClick(): void
#newSlaveConnected(m: Machine): void
#initiateAddBuild(b: Build): void
#slaveDisconnected(index: int): void
#on_actionStop_Server_triggered(): void
#on_actionSet_Port_triggered(): void
#on_actionStart_triggered(): void
#on_actionHelp_triggered(): void
#on_actionVersion_triggered(): void
#masterBuildsClicked(index: QModelIndex): void
#slaveStatsClicked(index: QModelIndex): void
#clearWidget(): void
#on_actionAdd_Build_triggered(): void
#displayBuilds(): void
#on_actionCopy_Build_Over_triggered(): void
#initiateCopyBuildOver(uniqueId: int, buildName: QString): void
#slaveGotBuild(m: Machine, buildId: int, slaveBuildName: QString, buildExist: bool): void
#slaveBuildSizeSame(buildId: int, machineId: int, isTheSame: bool): void
#slaveBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
#on_treeWidgetSlaves_clicked(index: QModelIndex): void
-dropBuildToSlave(from: QString): void
-dropNewBuildAdd(newBuildDirectory: QString): void
-displaySlaves(): void
-loadXMLBuilds(): void
-populateBuildInfo(retr: Build): void
-populateTreeWidgetInfo(retr: Build): void
-getSlaveTreeItemById(uniqueId: int): QTreeWidgetItem
Management
-server: Server
-allBuilds: Build
-buildCount: int
-machineCount: int
-allMachines: QVector<Machine*>
-lock: QMutex
-minStats: QString
-detStats: QString
<<CppMacro>>-()
-newSlaveConnected(m: Machine): void
-slaveDisconnected(index: int): void
-slaveGotBuild(machine: Machine, buildNo: int, buildName: QString, buildExists: bool): void
-slaveBuildSizeSame(buildId: int, slaveId: int, isTheSame: bool): void
-slaveBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
<<create>>-Management(parent: QObject)
<<destroy>>-Management()
+getBuildByID(id: int): Build
+getAllBuilds(): Build
+getBuildCount(): int
+getAllMachines(): QVector<Machine*>
+getMachineCount(): int
+addBuild(buildToAdd: Build): void
+addMachine(uniqueID: int, address: QString, handler: ProtocolHandler): void
+removeMachine(m: Machine): void
+getMachineAt(i: int): Machine
+setServer(serv: Server): void
+startServer(): void
+stopServer(): void
+setPort(newPort: int): void
+buildExistWithName(name: QString): bool
+getBuildByName(name: QString): Build
+machineExistWithIp(ip: QString): bool
+copyBuildOver(machineId: int, buildName: QString): void
+addBuildToSlave(machineId: int, buildNo: int, buildName: QString): void
+getBuildMD5(build: Build): QString
+getMachineById(machineId: int): Machine
+slaveBuildSize(buildNo: int, buildMD5Value: QString, slaveId: int): void
+setSlaveBuildIsSame(isSame: bool, machineId: int, buildID: int): void
+getMinStats(): QString
+setMinStats(stats: QString): void
+getDetStats(): QString
+setDetStats(stats: QString): void
+machineBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
-generateUniqueId(): int
management
CopyBuildOver
-ui: CopyBuildOver
-management: Management
-nameSuggestor: QCompleter
-ipSuggestor: QCompleter
<<CppMacro>>-()
<<create>>-CopyBuildOver(man: Management, suggestName: QStringList, suggestIP: QStringList, buildName: QString, parent: QWidget)
<<destroy>>-CopyBuildOver()
+cancelClick(): void
+okClick(): void
+buildMoreInformationClick(): void
+showMessage(errorMessage: QString, info: QString): void
+copyBuildOver(machineID: int, buildName: QString): void
-validateInput(): bool


MainForm
-ui: MainForm
-trayIcon: QSystemTrayIcon
-management: Management
-masterBuilds: MasterBuilds
-buildInfo: BuildInfo
-slaveStats: SlaveStats
<<CppMacro>>-()
-quitApplication(): void
<<create>>-MainForm(parent: QWidget)
<<destroy>>-MainForm()
#dropEvent(event: QDropEvent): void
#dragEnterEvent(event: QDragEnterEvent): void
#dragLeaveEvent(event: QDragLeaveEvent): void
#dragMoveEvent(event: QDragMoveEvent): void
#quitTheApplication(): void
#showOrHideTrayClick(): void
#newSlaveConnected(m: Machine): void
#initiateAddBuild(b: Build): void
#slaveDisconnected(index: int): void
#on_actionStop_Server_triggered(): void
#on_actionSet_Port_triggered(): void
#on_actionStart_triggered(): void
#on_actionHelp_triggered(): void
#on_actionVersion_triggered(): void
#masterBuildsClicked(index: QModelIndex): void
#slaveStatsClicked(index: QModelIndex): void
#clearWidget(): void
#on_actionAdd_Build_triggered(): void
#displayBuilds(): void
#on_actionCopy_Build_Over_triggered(): void
#initiateCopyBuildOver(uniqueId: int, buildName: QString): void
#slaveGotBuild(m: Machine, buildId: int, slaveBuildName: QString, buildExist: bool): void
#slaveBuildSizeSame(buildId: int, machineId: int, isTheSame: bool): void
#slaveBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
#on_treeWidgetSlaves_clicked(index: QModelIndex): void
-dropBuildToSlave(from: QString): void
-dropNewBuildAdd(newBuildDirectory: QString): void
-displaySlaves(): void
-loadXMLBuilds(): void
-populateBuildInfo(retr: Build): void
-populateTreeWidgetInfo(retr: Build): void
-getSlaveTreeItemById(uniqueId: int): QTreeWidgetItem
Server
-management: Management
-port: int
<<CppMacro>>-()
<<create>>-Server(m: Management, parent: QObject)
+startServer(): void
+stopServer(): void
+setPort(newPort: int): void
#incomingConnection(socketID: int): void
#showMessage(message: QString, flag: QString): void
<<creates>>
ServerThread
-socket: QTcpSocket
-socketID: int
-protocolHandler: ProtocolHandler
<<CppMacro>>-()
<<create>>-ServerThread(ID: int, man: Management, parent: QObject)
+run(): void
+getSocket(): QTcpSocket
+readyReadFunction(): void
+disconnectedFunction(): void
server
management
protocolHandler
Management
-server: Server
-allBuilds: Build
-buildCount: int
-machineCount: int
-allMachines: QVector<Machine*>
-lock: QMutex
-minStats: QString
-detStats: QString
<<CppMacro>>-()
-newSlaveConnected(m: Machine): void
-slaveDisconnected(index: int): void
-slaveGotBuild(machine: Machine, buildNo: int, buildName: QString, buildExists: bool): void
-slaveBuildSizeSame(buildId: int, slaveId: int, isTheSame: bool): void
-slaveBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
<<create>>-Management(parent: QObject)
<<destroy>>-Management()
+getBuildByID(id: int): Build
+getAllBuilds(): Build
+getBuildCount(): int
+getAllMachines(): QVector<Machine*>
+getMachineCount(): int
+addBuild(buildToAdd: Build): void
+addMachine(uniqueID: int, address: QString, handler: ProtocolHandler): void
+removeMachine(m: Machine): void
+getMachineAt(i: int): Machine
+setServer(serv: Server): void
+startServer(): void
+stopServer(): void
+setPort(newPort: int): void
+buildExistWithName(name: QString): bool
+getBuildByName(name: QString): Build
+machineExistWithIp(ip: QString): bool
+copyBuildOver(machineId: int, buildName: QString): void
+addBuildToSlave(machineId: int, buildNo: int, buildName: QString): void
+getBuildMD5(build: Build): QString
+getMachineById(machineId: int): Machine
+slaveBuildSize(buildNo: int, buildMD5Value: QString, slaveId: int): void
+setSlaveBuildIsSame(isSame: bool, machineId: int, buildID: int): void
+getMinStats(): QString
+setMinStats(stats: QString): void
+getDetStats(): QString
+setDetStats(stats: QString): void
+machineBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
-generateUniqueId(): int
ProtocolHandler
-slaveCurrentBuilds: Protocol
-copyOver: Protocol
-connect: Protocol
-sizeCheckBuilds: Protocol
-getSysInfo: Protocol
-sendBuild: Protocol
-management: Management
-firstTalk: bool
-machine: Machine
-slaveSocket: QTcpSocket
<<CppMacro>>-()
<<create>>-ProtocolHandler(man: Management, parent: QObject)
<<destroy>>-ProtocolHandler()
+setSocket(socket: QTcpSocket): void
+handle(data: QString): void
+setMachine(m: Machine): void
+getMachine(): Machine
+copyBuildOver(buildId: int, buildName: QString): void
+disconnectMachine(): void
+getMinStats(): void
+getDetStats(): void
-requestHandler(data: QString): void
+ContextInterface()

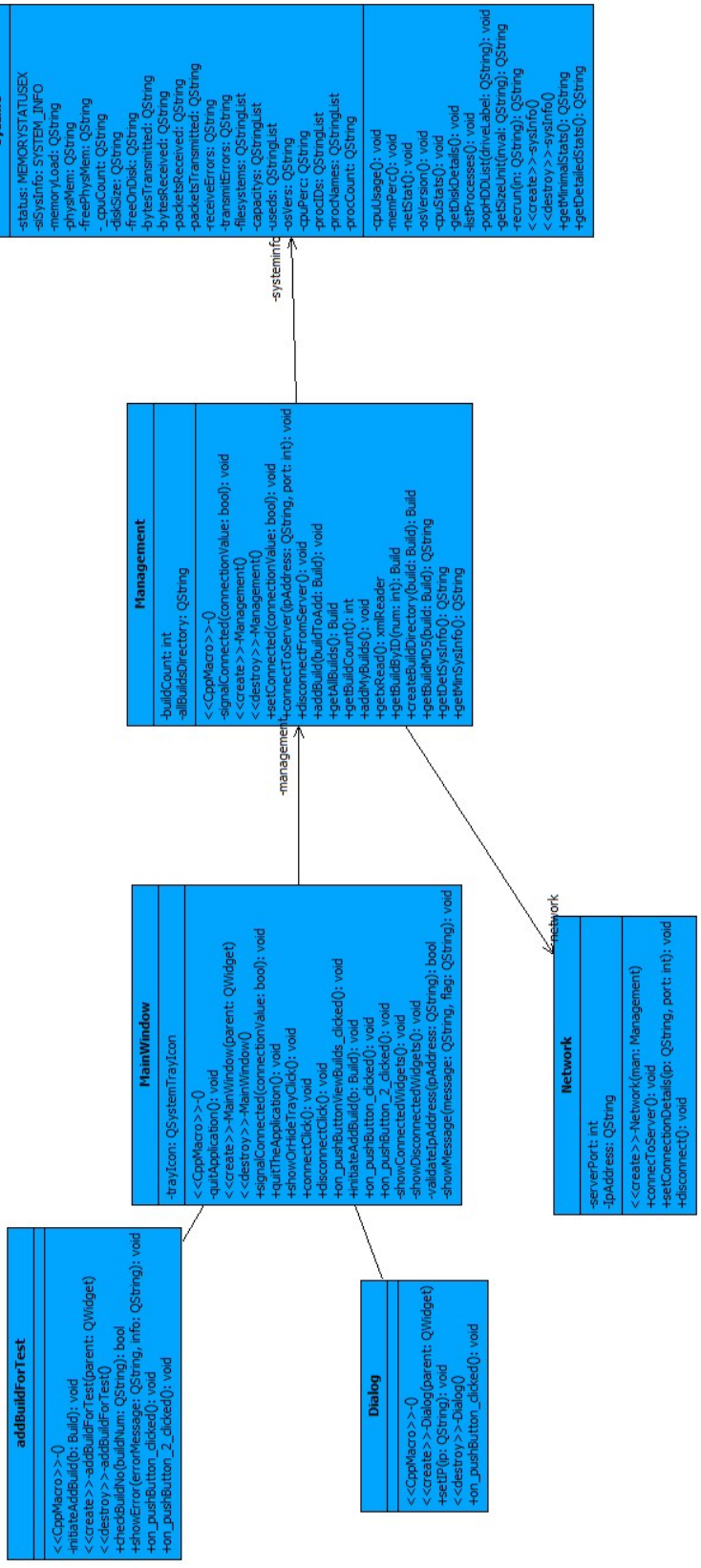

ProtocolHandler
-slaveCurrentBuilds: Protocol
-copyOver: Protocol
-connect: Protocol
-sizeCheckBuilds: Protocol
-getSysInfo: Protocol
-sendBuild: Protocol
-management: Management
-firstTalk: bool
-machine: Machine
-slaveSocket: QTcpSocket
<<CppMacro>>-()
<<create>>-ProtocolHandler(man: Management, parent: QObject)
<<destroy>>-ProtocolHandler()
+setSocket(socket: QTcpSocket): void
+handle(data: QString): void
+setMachine(m: Machine): void
+getMachine(): Machine
+copyBuildOver(buildId: int, buildName: QString): void
+disconnectMachine(): void
+getMinStats(): void
+getDetStats(): void
-requestHandler(data: QString): void
+ContextInterface()
+strategy
Protocol
<<CppMacro>>-()
<<create>>-Protocol(parent: QObject)
+handle(jsonObject: QVariantMap, man: Management, slaveSocket: QTcpSocket): void
#startJSONMessage(): QString
#appendJSONValue(currentString: QString, newKey: QString, newValue: QString, addComma: bool): void
#endJSONMessage(currentString: QString): void
+AlgorithmInterface()
ProtoConnect
<<create>>-ProtoConnect(parent: QObject)
+handle(jsonObject: QVariantMap, man: Management, slaveSocket: QTcpSocket): void
+disconnectMachine(machine: Machine, management: Management): void
ProtoCopyOver
<<create>>-ProtoCopyOver(parent: QObject)
+handle(jsonObject: QVariantMap, man: Management, slaveSocket: QTcpSocket): void
+copyBuildOver(buildId: int, buildName: QString, slaveSocket: QTcpSocket): void
-GotABuild(jsonObject: QVariantMap, man: Management, slaveSocket: QTcpSocket): void
-sizeCheckCertainBuild(buildNo: QString, slaveSocket: QTcpSocket): void
ProtoSendBuild
<<create>>-ProtoSendBuild(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
+SizeCheckAllBuildsDone(slaveSocket: QTcpSocket, management: Management): void
+sizeCheckCertainBuildDone(): void
ProtoSlaveCurrentBuilds
<<create>>-ProtoSlaveCurrentBuilds(parent: QObject)
+handle(jsonObject: QVariantMap, man: Management, slaveSocket: QTcpSocket): void
-RecheckDone(slaveSocket: QTcpSocket): void
-RecheckBuilds(slaveSocket: QTcpSocket): void
-Rechecker(jsonObject: QVariantMap, management: Management, socket: QTcpSocket): void
-SizeCheckAllBuilds(slaveSocket: QTcpSocket): void
ProtoSizeCheckBuilds
<<create>>-ProtoSizeCheckBuilds(parent: QObject)
+handle(jsonObject: QVariantMap, man: Management, slaveSocket: QTcpSocket): void
-BuildMD5(jsonObject: QVariantMap, management: Management, slaveSocket: QTcpSocket): void

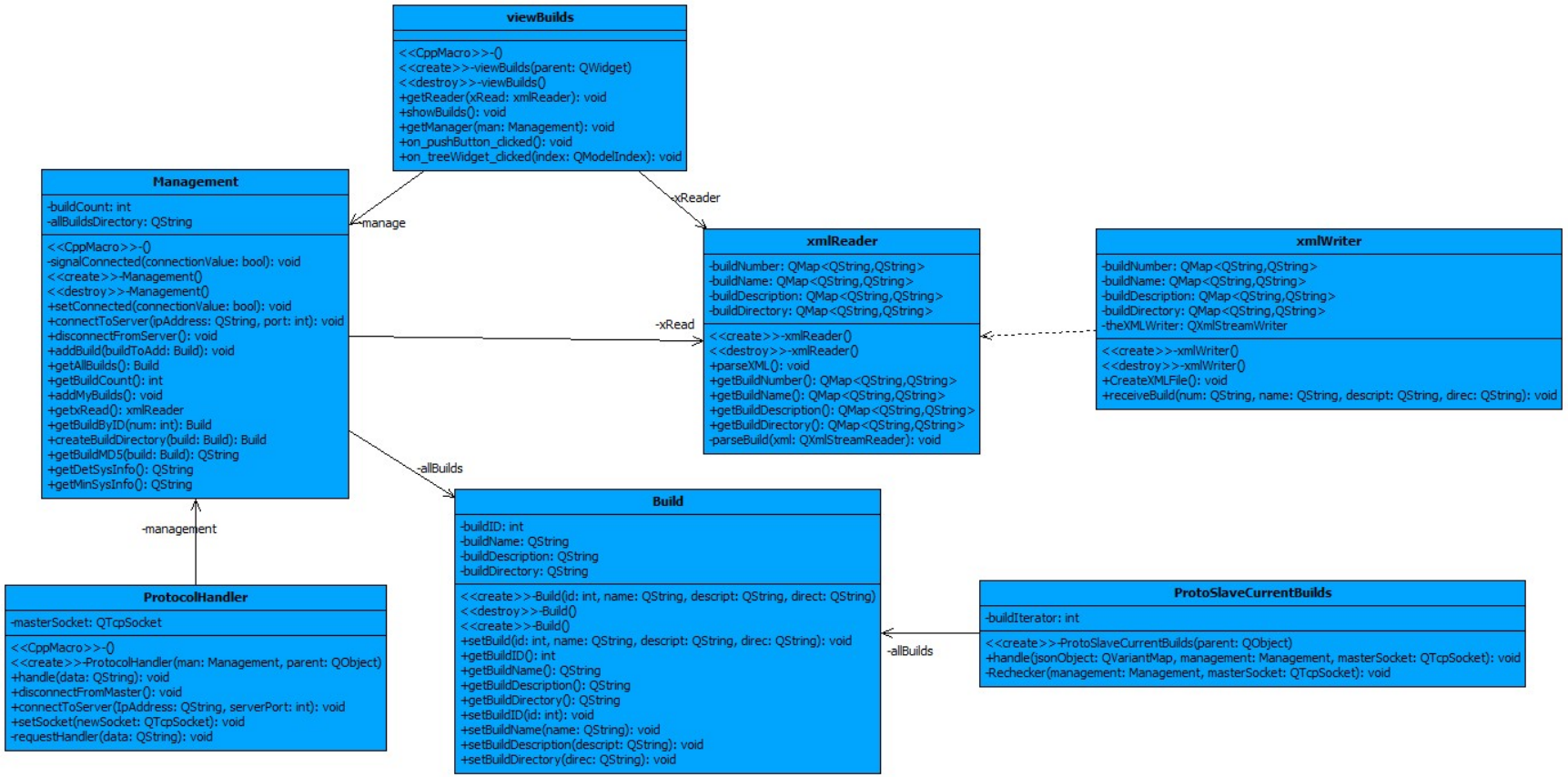

ProtoSendBuild
<<create>>-ProtoSendBuild(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
+SizeCheckAllBuildsDone(slaveSocket: QTcpSocket, management: Management): void
+sizeCheckCertainBuildDone(): void
compression
<<create>>-compression()
+compress(dirs: QStringList, toDir: QString, name: QString): void
+decompress(fromDir: QString, toDir: QString, name: QString): void
CopySenderServer
-ipAddress: QString
-differentBuildDirectories: QStringList
-differentBuildNos: QStringList
-socket: QTcpSocket
-firstTalk: bool
-management: Management
-machineId: int
<<CppMacro>>-()
<<create>>-CopySenderServer(diffBuilds: QStringList, diffBuildNos: QStringList, man: Management, mashId: int, parent: QObject)
<<destroy>>-CopySenderServer()
+startServer(): int
+stopServer(): void
+destroyServer(): void
+readyReadFunction(): void
-handle(data: QString): void
-requestHandler(data: QString): void
-SendDifferences(): void
-getDifferences(jsonObject: QVariantMap): void
-createCopyCompare(keys: QList<QString>, allMD5s: QVariantMap, md5Class: BuildMD5, theBuildDirectory: QString): CopyCompare
#incomingConnection(socketID: int): void
BuildMD5
-buildFiles: QStringList
-buildFilesMD5: QStringList
-currentIndex: int
<<CppMacro>>-()
<<create>>-BuildMD5(parent: QObject)
<<destroy>>-BuildMD5()
+generateAllMD5(directory: QString): void
+getBuildFiles(): QStringList
+getBuildFilesMD5(): QStringList
+getCurrentBuildDirectory(): QString
+getCurrentFileMd5Sum(): QString
+next(): void
+getCurrentIndex(): int
+getSize(): int
CopyCompare
-numFilesNotCopied: unsigned long int
-filePaths: QStringList
-deleteFilePaths: QStringList
<<CppMacro>>-()
<<create>>-CopyCompare(parent: QObject)
<<destroy>>-CopyCompare()
+compareMD5(masterMD5Value: QString, masterFilePath: QString, slaveMd5Value: QString): void
+createFile(fileDirectory: QString): void
+percentageSynched(): double
+deleteFile(file: QString): void
+getDeleteJsonString(buildNo: QString): QString

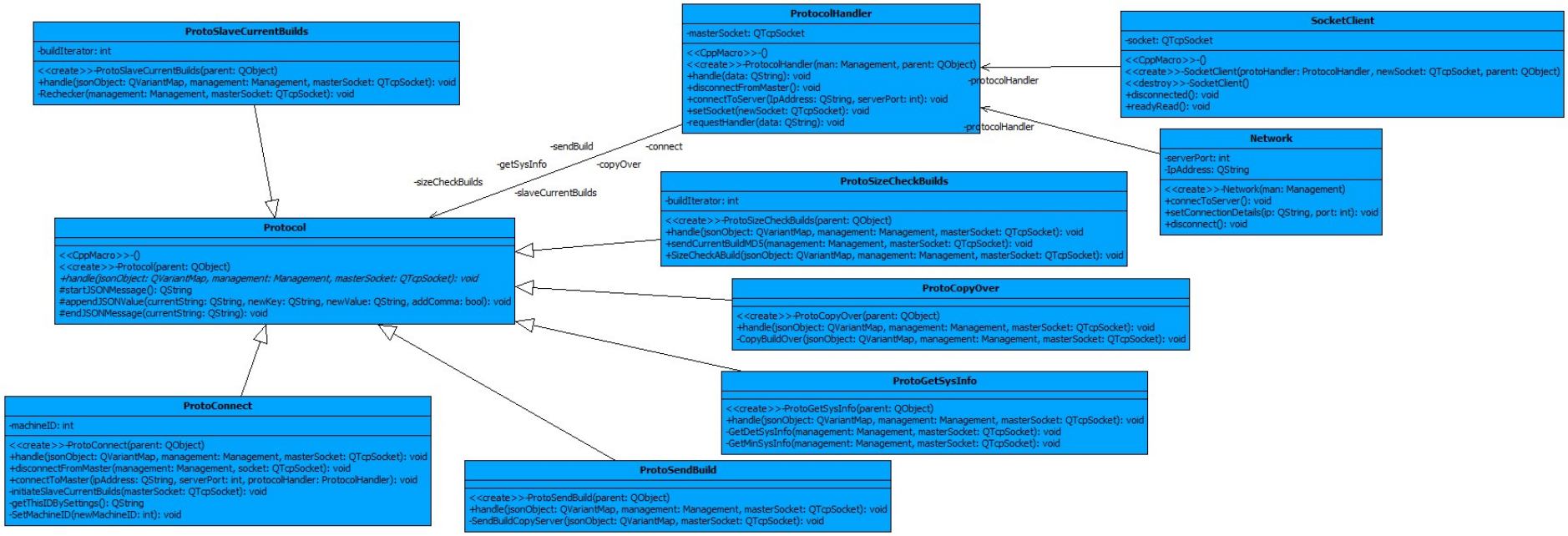

Management
-server: Server
-allBuilds: Build
-buildCount: int
-machineCount: int
-allMachines: QVector<Machine*>
-lock: QMutex
-minStats: QString
-detStats: QString
<<CppMacro>>-()
-newSlaveConnected(m: Machine): void
-slaveDisconnected(index: int): void
-slaveGotBuild(machine: Machine, buildNo: int, buildName: QString, buildExists: bool): void
-slaveBuildSizeSame(buildId: int, slaveId: int, isTheSame: bool): void
-slaveBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
<<create>>-Management(parent: QObject)
<<destroy>>-Management()
+getBuildByID(id: int): Build
+getAllBuilds(): Build
+getBuildCount(): int
+getAllMachines(): QVector<Machine*>
+getMachineCount(): int
+addBuild(buildToAdd: Build): void
+addMachine(uniqueID: int, address: QString, handler: ProtocolHandler): void
+removeMachine(m: Machine): void
+getMachineAt(i: int): Machine
+setServer(serv: Server): void
+startServer(): void
+stopServer(): void
+setPort(newPort: int): void
+buildExistWithName(name: QString): bool
+getBuildByName(name: QString): Build
+machineExistWithIp(ip: QString): bool
+copyBuildOver(machineId: int, buildName: QString): void
+addBuildToSlave(machineId: int, buildNo: int, buildName: QString): void
+getBuildMD5(build: Build): QString
+getMachineById(machineId: int): Machine
+slaveBuildSize(buildNo: int, buildMD5Value: QString, slaveId: int): void
+setSlaveBuildIsSame(isSame: bool, machineId: int, buildID: int): void
+getMinStats(): QString
+setMinStats(stats: QString): void
+getDetStats(): QString
+setDetStats(stats: QString): void
+machineBuildSynched(machineId: int, buildId: int, percentageSynched: double): void
-generateUniqueId(): int
allMachines
Machine
#machineID: int
#machineIP: QString
<<create>>-Machine(id: int, ip: QString)
<<destroy>>-Machine()
<<create>>-Machine()
+setMachine(id: int, ip: QString): void
+getMachineID(): int
+getMachineIP(): QString
+setMachineID(id: int): void
+setMachineIP(ip: QString): void
+deleteBuild(name: QString): void
+deleteBuild(id: int): void
+addBuild(buildToAdd: Build): void
+setProtocolHandler(handler: ProtocolHandler): void
+copyBuildOver(build: Build): void
+getBuilds(): Build
+getBuildCount(): int
+setBuildSame(isSame: bool, buildID: int): void
+getDetStats(): void
+getMinStats(): void
ProtocolHandler
-slaveCurrentBuilds: Protocol
-copyOver: Protocol
-connect: Protocol
-sizeCheckBuilds: Protocol
-getSysInfo: Protocol
-sendBuild: Protocol
-management: Management
-firstTalk: bool
-machine: Machine
-slaveSocket: QTcpSocket
<<CppMacro>>-()
<<create>>-ProtocolHandler(man: Management, parent: QObject)
<<destroy>>-ProtocolHandler()
+setSocket(socket: QTcpSocket): void
+handle(data: QString): void
+setMachine(m: Machine): void
+getMachine(): Machine
+copyBuildOver(buildId: int, buildName: QString): void
+disconnectMachine(): void
+getMinStats(): void
+getDetStats(): void
-requestHandler(data: QString): void
+ContextInterface()
allBuilds
protocolHandler
Build
-buildID: int
-buildName: QString
-buildDescription: QString
-buildDirectory: QString
-isSame: bool
<<create>>-Build(id: int, name: QString, descript: QString, direct: QString)
<<destroy>>-Build()
<<create>>-Build()
+setBuild(id: int, name: QString, descript: QString, direc: QString): void
+getBuildID(): int
+getBuildName(): QString
+getBuildDescription(): QString
+getBuildDirectory(): QString
+setBuildID(id: int): void
+setBuildName(name: QString): void
+setBuildDescription(descript: QString): void
+setBuildDirectory(direc: QString): void
+setIsSame(same: bool): void
+getIsSame(): bool
slaveBuilds
Slave
-slaveBuilds: Build
-buildCount: int
-protocolHandler: ProtocolHandler
<<create>>-Slave(id: int, ip: QString)
<<destroy>>-Slave()
<<create>>-Slave()
<<create>>-Slave(toCopy: Slave)
+deleteBuild(name: QString): void
+deleteBuild(id: int): void
+addBuild(buildToAdd: Build): void
+setProtocolHandler(handler: ProtocolHandler): void
+copyBuildOver(b: Build): void
+getBuilds(): Build
+getBuildCount(): int
+setBuildSame(isSame: bool, buildID: int): void
+getMinStats(): void
+getDetStats(): void

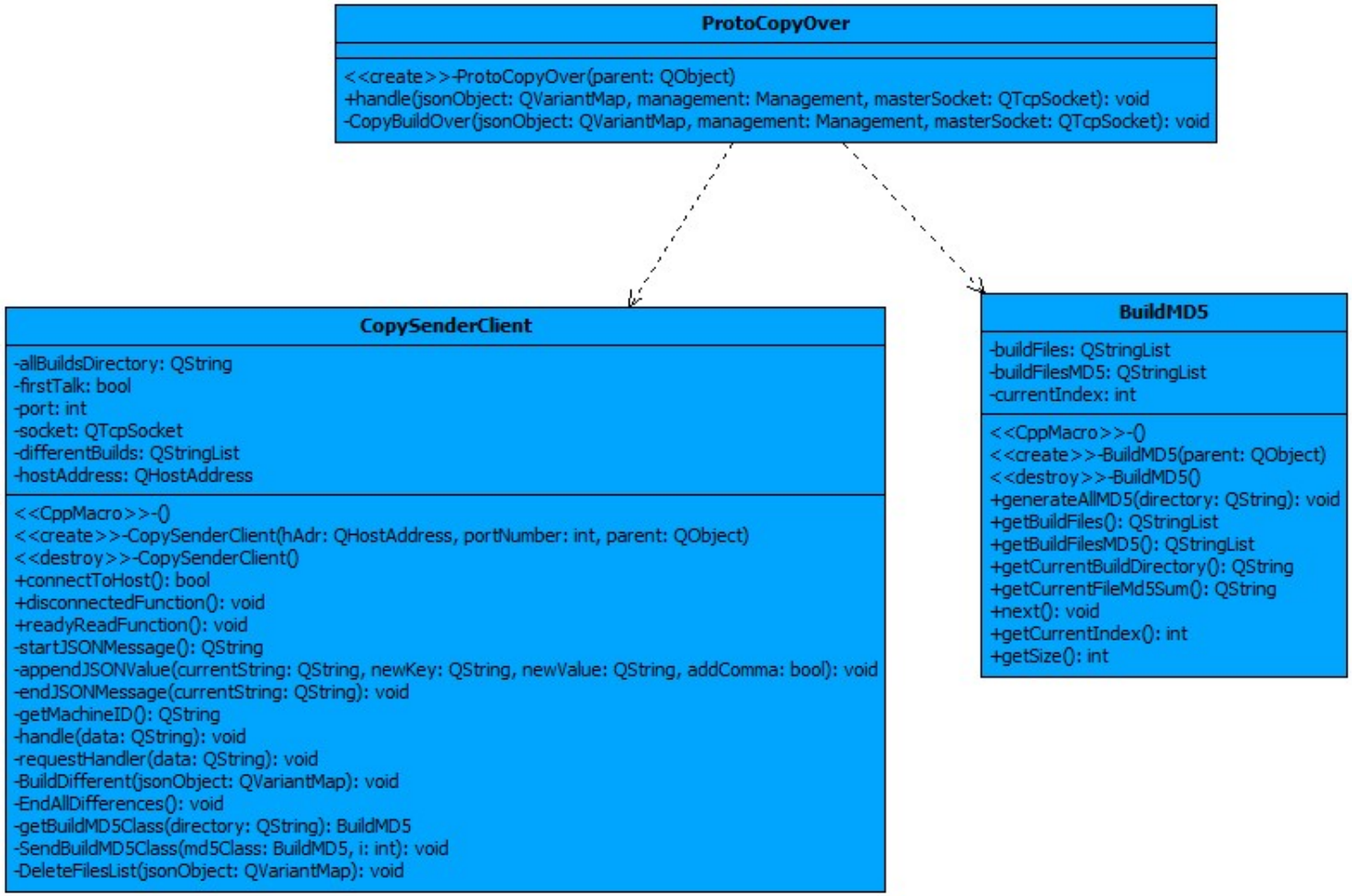
4.2 AppManClient Class Diagram

Due to its large size, the class diagram is split into multiple, seperate diagrams. Classes will be repeated to show how they all link.




viewBuilds
<<CppMacro>>-()
<<create>>-viewBuilds(parent: QWidget)
<<destroy>>-viewBuilds()
+getReader(xRead: xmlReader): void
+showBuilds(): void
+getManager(man: Management): void
+on_pushButton_clicked(): void
+on_treeWidget_clicked(index: QModelIndex): void
Management
-buildCount: int
-allBuildsDirectory: QString
<<CppMacro>>-()
-signalConnected(connectionValue: bool): void
<<create>>-Management()
<<destroy>>-Management()
+setConnected(connectionValue: bool): void
+connectToServer(ipAddress: QString, port: int): void
+disconnectFromServer(): void
+addBuild(buildToAdd: Build): void
+getAllBuilds(): Build
+getBuildCount(): int
+addMyBuilds(): void
+getxRead(): xmlReader
+getBuildByID(num: int): Build
+createBuildDirectory(build: Build): Build
+getBuildMD5(build: Build): QString
+getDetSysInfo(): QString
+getMinSysInfo(): QString
manage
xReader
xmlReader
-buildNumber: QMap<QString,QString>
-buildName: QMap<QString,QString>
-buildDescription: QMap<QString,QString>
-buildDirectory: QMap<QString,QString>
<<create>>-xmlReader()
<<destroy>>-xmlReader()
+parseXML(): void
+getBuildNumber(): QMap<QString,QString>
+getBuildName(): QMap<QString,QString>
+getBuildDescription(): QMap<QString,QString>
+getBuildDirectory(): QMap<QString,QString>
-parseBuild(xml: QXmlStreamReader): void
xmlWriter
-buildNumber: QMap<QString,QString>
-buildName: QMap<QString,QString>
-buildDescription: QMap<QString,QString>
-buildDirectory: QMap<QString,QString>
-theXMLWriter: QXmlStreamWriter
<<create>>-xmlWriter()
<<destroy>>-xmlWriter()
+CreateXMLFile(): void
+receiveBuild(num: QString, name: QString, descript: QString, direc: QString): void
-xRead
-allBuilds
-management
ProtocolHandler
-masterSocket: QTcpSocket
<<CppMacro>>-()
<<create>>-ProtocolHandler(man: Management, parent: QObject)
+handle(data: QString): void
+disconnectFromMaster(): void
+connectToServer(IpAddress: QString, serverPort: int): void
+setSocket(newSocket: QTcpSocket): void
-requestHandler(data: QString): void
Build
-buildID: int
-buildName: QString
-buildDescription: QString
-buildDirectory: QString
<<create>>-Build(id: int, name: QString, descript: QString, direct: QString)
<<destroy>>-Build()
<<create>>-Build()
+setBuild(id: int, name: QString, descript: QString, direc: QString): void
+getBuildID(): int
+getBuildName(): QString
+getBuildDescription(): QString
+getBuildDirectory(): QString
+setBuildID(id: int): void
+setBuildName(name: QString): void
+setBuildDescription(descript: QString): void
+setBuildDirectory(direc: QString): void
ProtoSlaveCurrentBuilds
-buildIterator: int
<<create>>-ProtoSlaveCurrentBuilds(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
-Rechecker(management: Management, masterSocket: QTcpSocket): void
-allBuilds

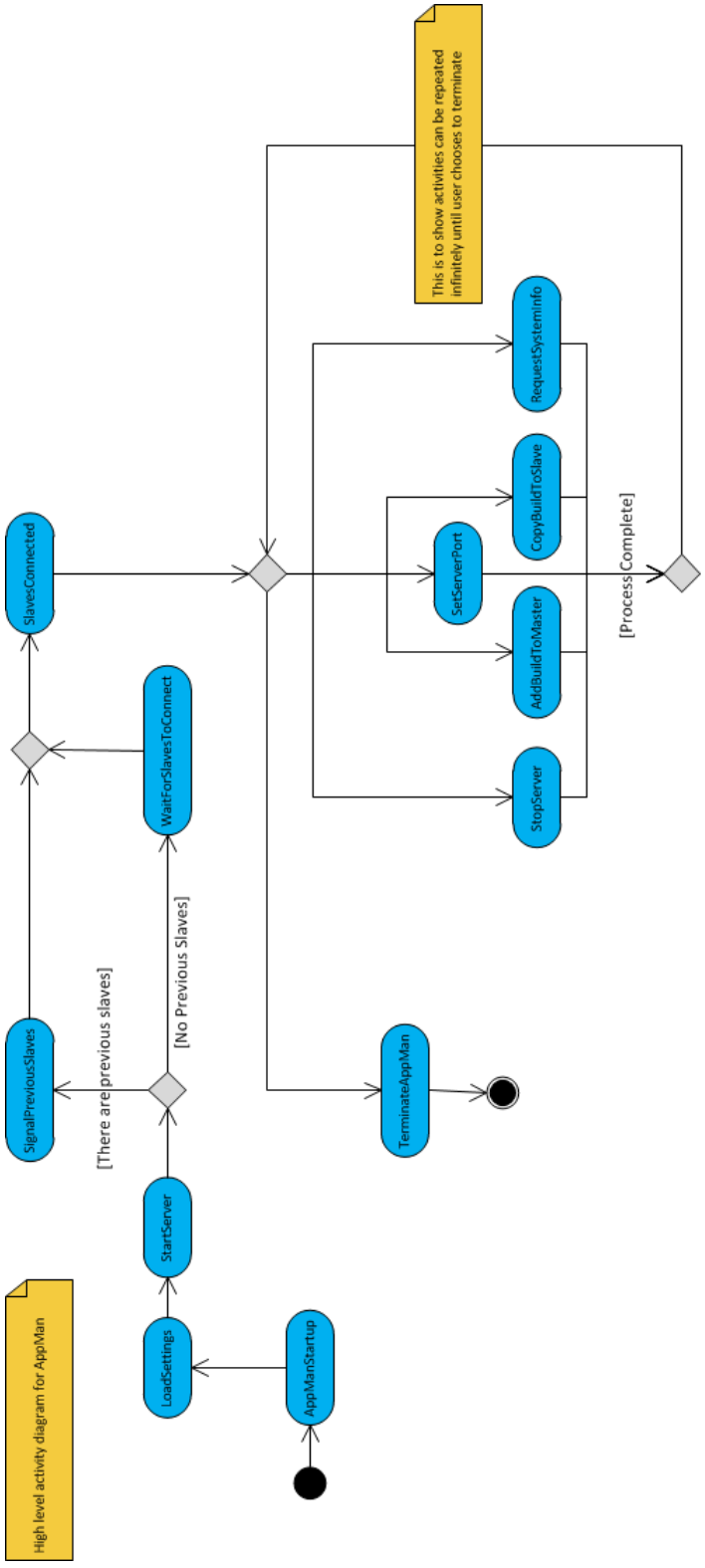

ProtoSlaveCurrentBuilds
-buildIterator: int
<<create>>-ProtoSlaveCurrentBuilds(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
-Rechecker(management: Management, masterSocket: QTcpSocket): void
ProtocolHandler
-masterSocket: QTcpSocket
<<CppMacro>>-()
<<create>>-ProtocolHandler(man: Management, parent: QObject)
+handle(data: QString): void
+disconnectFromMaster(): void
+connectToServer(IpAddress: QString, serverPort: int): void
+setSocket(newSocket: QTcpSocket): void
-requestHandler(data: QString): void
-protocolHandler
-protocolHandler
SocketClient
-socket: QTcpSocket
<<CppMacro>>-()
<<create>>-SocketClient(protoHandler: ProtocolHandler, newSocket: QTcpSocket, parent: QObject)
<<destroy>>-SocketClient()
+disconnected(): void
+readyRead(): void
Network
-serverPort: int
-IpAddress: QString
<<create>>-Network(man: Management)
+connecToServer(): void
+setConnectionDetails(ip: QString, port: int): void
+disconnect(): void
-sendBuild
-connect
-getSysInfo
-copyOver
-sizeCheckBuilds
-slaveCurrentBuilds
Protocol
<<CppMacro>>-()
<<create>>-Protocol(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
#startJSONMessage(): QString
#appendJSONValue(currentString: QString, newKey: QString, newValue: QString, addComma: bool): void
#endJSONMessage(currentString: QString): void
ProtoSizeCheckBuilds
-buildIterator: int
<<create>>-ProtoSizeCheckBuilds(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
+sendCurrentBuildMD5(management: Management, masterSocket: QTcpSocket): void
+SizeCheckABuild(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
ProtoCopyOver
<<create>>-ProtoCopyOver(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
-CopyBuildOver(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
ProtoGetSysInfo
<<create>>-ProtoGetSysInfo(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
-GetDetSysInfo(management: Management, masterSocket: QTcpSocket): void
-GetMinSysInfo(management: Management, masterSocket: QTcpSocket): void
ProtoConnect
-machineID: int
<<create>>-ProtoConnect(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
+disconnectFromMaster(management: Management, socket: QTcpSocket): void
+connectToMaster(ipAddress: QString, serverPort: int, protocolHandler: ProtocolHandler): void
-initiateSlaveCurrentBuilds(masterSocket: QTcpSocket): void
-getThisIDBySettings(): QString
-SetMachineID(newMachineID: int): void
ProtoSendBuild
<<create>>-ProtoSendBuild(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
-SendBuildCopyServer(jsonObject: QVariantMap, masterSocket: QTcpSocket): void


ProtoCopyOver
<<create>>-ProtoCopyOver(parent: QObject)
+handle(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
-CopyBuildOver(jsonObject: QVariantMap, management: Management, masterSocket: QTcpSocket): void
CopySenderClient
-allBuildsDirectory: QString
-firstTalk: bool
-port: int
-socket: QTcpSocket
-differentBuilds: QStringList
-hostAddress: QHostAddress
<<CppMacro>>-()
<<create>>-CopySenderClient(hAdr: QHostAddress, portNumber: int, parent: QObject)
<<destroy>>-CopySenderClient()
+connectToHost(): bool
+disconnectedFunction(): void
+readyReadFunction(): void
-startJSONMessage(): QString
-appendJSONValue(currentString: QString, newKey: QString, newValue: QString, addComma: bool): void
-endJSONMessage(currentString: QString): void
-getMachineID(): QString
-handle(data: QString): void
-requestHandler(data: QString): void
-BuildDifferent(jsonObject: QVariantMap): void
-EndAllDifferences(): void
-getBuildMD5Class(directory: QString): BuildMD5
-SendBuildMD5Class(md5Class: BuildMD5, i: int): void
-DeleteFilesList(jsonObject: QVariantMap): void
BuildMD5
-buildFiles: QStringList
-buildFilesMD5: QStringList
-currentIndex: int
<<CppMacro>>-()
<<create>>-BuildMD5(parent: QObject)
<<destroy>>-BuildMD5()
+generateAllMD5(directory: QString): void
+getBuildFiles(): QStringList
+getBuildFilesMD5(): QStringList
+getCurrentBuildDirectory(): QString
+getCurrentFileMd5Sum(): QString
+next(): void
+getCurrentIndex(): int
+getSize(): int

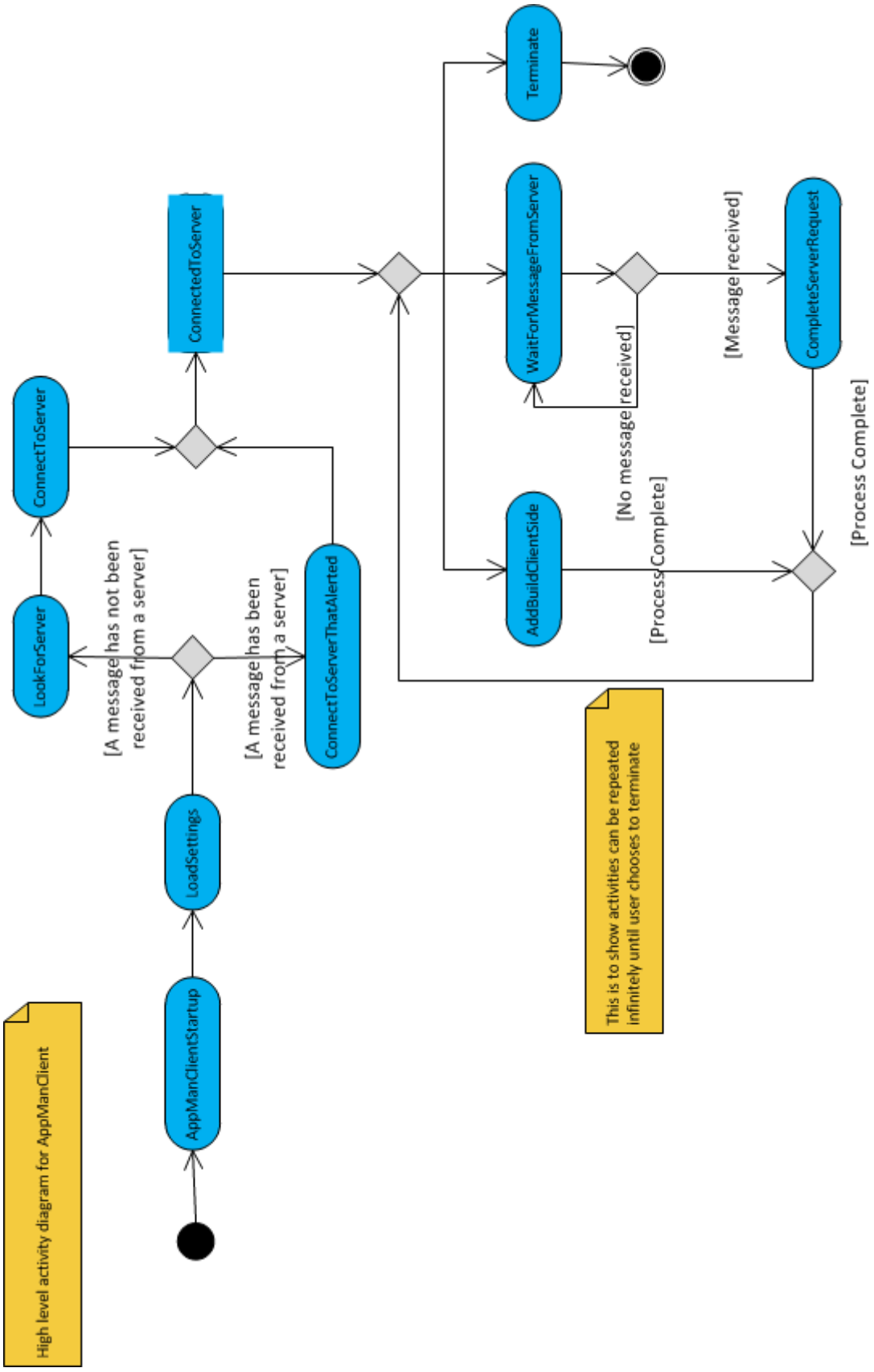
5 Overall Processes

This section is for the Activity diagrams based on current progress of AppMan and AppManClient respectively.

5.1 AppMan



5.2 AppManClient



6 Communication Protocol

7 Glossary

- Build - An application build version that could potentially be distributed to slave computers.
- Slave - A computer that will be controlled via a master computer. Application builds will be sent to this computer.
- Master - A computer that will control Slaves across a network.
- Server - A machine waiting on the network for connections from other machines.
- GUI - Graphical User Interface with which a user can control the project.
- Project - This project. The distributed application manager.
- Application Configuration - Environment variables that are specified when running an application.