Lab of Things:
Remote Update and Configuration

The configuration (or Config) consists of a set of XML files that govern the run-time behavior of a HomeOS instance (Home Hub). Config defines which sensors/devices are associated with the Home Hub, which drivers and apps are running, which users have access to the Home Hub and to which apps, and many other parameters. All of the configuration files for a Home Hub are stored in \Hub\output\Configs (each subdirectory contains a set of config files that defines a unique configuration).

Home Hubs are designed to frequently (every 5 minutes) interface with a Windows Azure Blobstore to report a part of their Config and apply updates to their Config (if any). By default, a Home Hub reports the modules, scouts, services, and access rules it's running under (stored in the Scouts.xml, Modules.xml, Services.xml, and Rules.xml files in the Config folder, respectively). This ensures that an experimenter accesses no user-specific private data (e.g., WiFi password, stored under Settings.xml).

Contents

Managing Configuration Files

This section discusses modifying configurations for your Home Hubs.

Config Packager Tool

With the config packager tool, you can download the configuration files for one or more Home Hubs, and also update the configs of Home Hubs.

To View a Hub's Latest Config

1. Open an elevated command prompt.
2. Navigate to Hub\Tools\ConfigPackager\bin\Debug.
3. Run ConfigPackager.exe as shown in the following example:

```
ConfigPackager.exe  --Function getactual --AccountName <AzureStorageAccount>
--AccountKey <AzureStorageAccountKey> --ActualConfigDir <LocalDirectory>
```

   where `<LocalDirectory>` is the directory where the downloaded Configs for the Hubs are to be stored. By default, the organization ID and study ID are set to 'default' and can be specified using `--OrgID` and `--StudyID`. By default, the tools obtains Configs of all hubs of a given (or default) organization ID and study ID, specific Hub identifiers for obtaining specific Configs can be specified using --HomeIDs <comma separated Hub IDs>. If needed, the storage account key can be read from a file specified using `--AccountKeyFile`. The Azure storage container on which the Configs can be specified using `--Container`, and defaults to 'configs'.

4. The Configs will be stored under <LocalDirectory>\<OrgID>\<StudyID>.

To Update Home Hub Configs

1. Open an elevated command prompt.
2. Navigate to Hub\Tools\ConfigPackager\bin\Debug.
3. Run ConfigPackager.exe as shown in the following example:

```
ConfigPackager.exe  --Function getactual --AccountName <AzureStorageAccount>
--AccountKey <AzureStorageAccountKey> --ActualConfigDir .\hubConfigs
```

4. The resulting directory will contain a directory named <OrgID>, which contains a directory named <StudyID>. This is where the Hubs' config files are located.
5. Make a copy of the directory (e.g. .\hubConfigs-Current). You can use the following command:

```
xcopy .\hubConfigs .\hubConfigs-Current /E
```

6. Edit the Hub config files stored in hubConfigs\<OrgID>\<StudyID>.
   - By changing the config XML files, you can start and stop modules and scouts on the hub.
   - The .versiondef file in Config contains (as a semicolon-delimited list) which parts of the Config are reported by the hubs during their periodic updates. By default the hubs only report Modules.xml, Scouts.xml, Services.xml and Rules.xml. You may add other config files to this list as required.
   - Please **DO NOT** modify the .currentversion and .parentversion files. Thes are used by hubs for version control of Configs, and resolving Config update conflicts between experimenters and participants.
7. Deploy the modified config on the Blob store by running the following command:

   By default, the organization ID and study ID are set to 'default' and can be specified using `--OrgID` and `--StudyID`. By default, the tools deploys the Configs of all hubs of a given (or

```
ConfigPackager.exe  --Function setdesired --AccountName <AzureStorageAccount>
--AccountKey <AzureStorageAccountKey> --ActualConfigDir .\hubConfigs-Current
--DesiredConfigDir .\hubConfigs
```

default) organization ID and study ID, present in `<DesiredConfigDir>`; specific Hub identifiers for deploying specific Configs can be specified using `--HomeIDs` <comma separated Hub IDs>. If needed, the storage account key can be read from a file specified using `--AccountKeyFile`. The Azure storage container on which the Configs can be specified using `--Container`, and defaults to 'configs'.

Deployment of the Config on the Hub can be confirmed by periodically monitoring the current Hub Config, as explained in To View a Hub's Latest Config. See note 2 below.

Notes

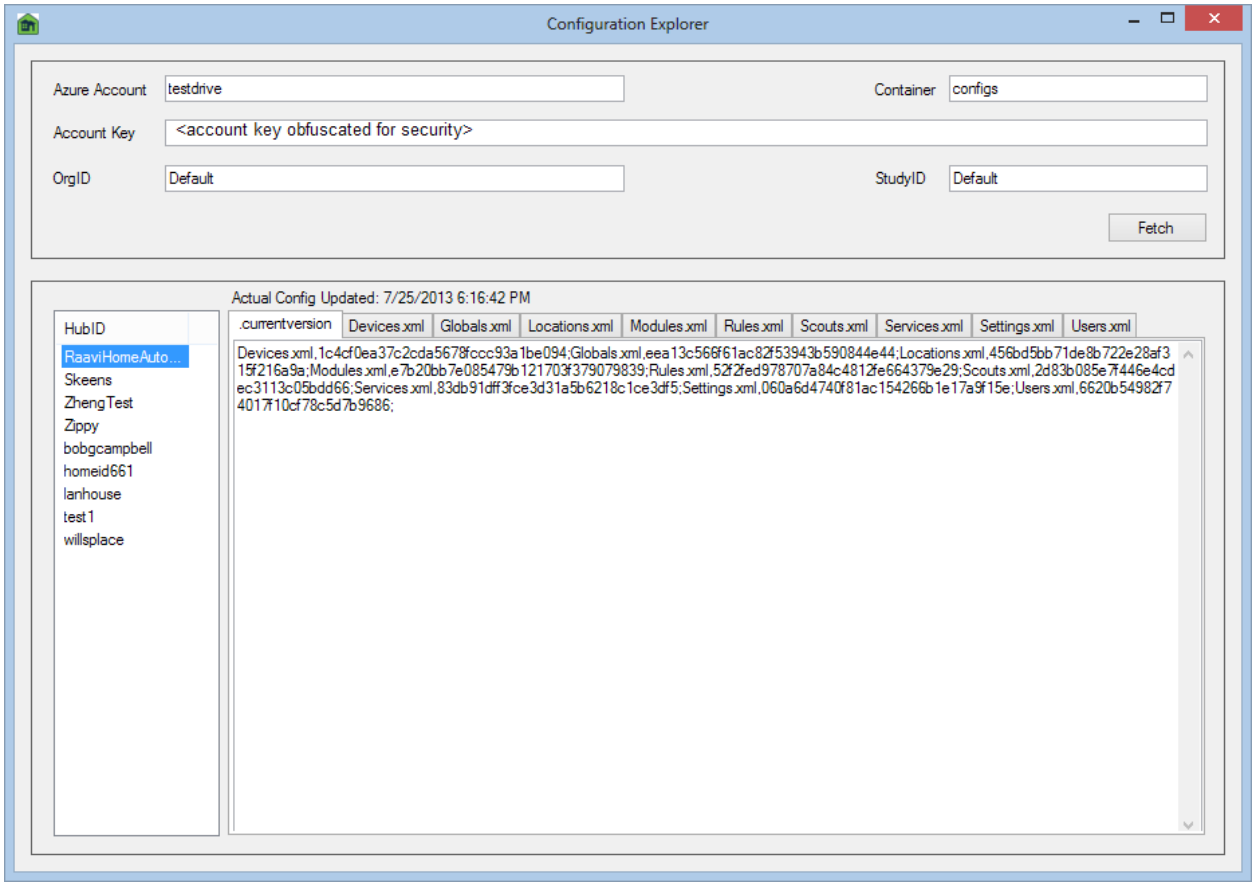
1. Use `--Help` to get a list of all commands and their meanings.
2. Deploying the Config on the blob store may not result in the Config being actually deployed on the Hub, as the Hub's current Config may change during the time its Config was being updated/edited. In this scenario, the experimenter shall need to re-do the steps in To Update Home Hub Configs until the hub applies the updates (verified by periodically viewing the current Hub Config, as explained in To View a Hub's Latest Config).

Configuration Explorer Tool

You can use the Configuration Explorer tool to browse the configuration files for your hubs, as stored in an Azure blob.

To run Configuration Explorer

1. Launch `homeos2\Hub\Tools\ConfigExplorer\bin\Debug\WindowsFormsApplication1.exe`.
2. In the **Azure Account** field, enter the name of the Azure blob storage account to use.
3. In the **Account Key** field, enter the Azure account key.
4. Click **Fetch**.
5. Under HubID, select a hub to view config data.



Configuration Files Reference

Devices.xml

Lists all devices using the following syntax:

```
<Device Id="myDeviceId" Driver="myDeviceDriver"/>
```

Globals.xml

[]

Locations.xml

Lists the physical locations for a study using the following syntax:

```
<Location Name="Home">
  <Location Name="Mobile"/>
  <Location Name="Outdoors"/>
  <Location Name="Indoors">
    <Location Name="Living room"/>
  </Location>
</Location>
```

Modules.xml

Lists the modules that have been added. The following example shows the contents of Modules.xml for the "dummy" configuration:

```
<Modules>
  <Module FriendlyName="DriverDummy" AppName="DriverDummy"
BinaryName="HomeOS.Hub.Drivers.Dummy" AutoStart="1" Background="1" Version="1.0.0.0">
    <Args Count="1" val1="Hero"/>
  </Module>

  <Module FriendlyName="AppDummy" AppName="AppDummy"
BinaryName="HomeOS.Hub.Apps.Dummy" AutoStart="1" Version="1.0.0.0">
    <Args Count="1" val1="Zero"/>
    <RoleList>
      <Role Name=":dummy:"/>
    </RoleList>
  </Module>
</Modules>
```

Rules.xml

Defines rules to control access for modules. The following example shows the contents of Rules.xml for the "dummy" configuration:

```
<Rules>

  <Rule Name="GuiWebRule" Module="GuiWeb">
    <Group Name="Everyone" Type="Allow" />
  </Rule>
  <Rule Name="ScoutRule" Module="scouts">
    <Group Name="Everyone" Type="Allow" />
  </Rule>
  <!-- Sample Rule to make this app AppDummy1 available to all users in users.xml
(i.e. including systemlow and systemhigh users)

  Rule Name="MyAppRule1" Module="AppDummy1">
    <Group Name="Everyone" Type="Allow" />
  </Rule-->

  <!-- Sample Rule to make this app AppDummy1 available to jane  on Sundays from 6pm
to 8pm
  Note: system high can access this app at all times.
   <Rule Name="MyAppRule2" Module="AppDummy1">
    <User Name="jane" Type="Allow">
      <Time DayOfWeek="0" StartMins="1800" EndMins="2000"/>
    </User>
  </Rule>
  -->
</Rules>
```

Scouts.xml

Defines the scouts to run. The following example shows the contents of Scouts.xml for the "dummy" configuration.

```
<Scouts>
  <Scout Name="HomeOS.Hub.Scouts.WebCam" DllName="HomeOS.Hub.Scouts.WebCam.dll"/>
  <Scout Name="HomeOS.Hub.Scouts.Foscam" DllName="HomeOS.Hub.Scouts.Foscam.dll"/>
  <Scout Name="HomeOS.Hub.Scouts.Gadgeteer"
DllName="HomeOS.Hub.Scouts.Gadgeteer.dll"/>
</Scouts>
```

Services.xml

Defines the services to use. The following example shows the contents of Services.xml for the "dummy" configuration:

```
<Services>
  <Service FriendlyName="port - DriverDummy" Module="DriverDummy"
ModuleFacingName="port" Location="Home" Configured="no">
    <Role Name=":dummy:" />
  </Service>
</Services>
```

Settings.xml

Defines the settings for a study. This is the most critical of all settings files. The following table describes each entry.

Entry	**Description**
OrgId	Organization ID for the study.
StudyId	ID of the study itself.
HomeId	ID of the home where this particular hub is located.
WifiSsid	SSID for the wireless network to use.
WifiKey	Security key for the wireless network.
ConfigDir	Directory for the config to use.
ModuleWorkingDirBase	Working directory for use by modules.
RunningMode	Select which running mode to use.
StayOffline	Prevents platform from going online if set to True.
EnforcePolicies	???
LogFile	Logfile to use.
LogArchivalDir	Directory to use for archival log storage.
LogRotationThreshold	Time between log updates in milliseconds.
AutoSyncLogs	Set whether to automatically synchronize log files.
PortRegisterDelay	???
MaxStopExecutionTime	???
MaxFinallyBlockExecutionTime	???
HomeStoreBase	Base directory for Home Store files.
RepositoryURIs	Base directory for Home Store repository URIs.
GatekeeperURI	URI for the gatekeeper.
HeartbeatServiceHost	Host name for the heartbeat service.
DataStoreAccountName	Name of the Windows Azure blob storage account to use.
DataStoreAccountKey	Key for the Windows Azure blob storage account.
ConfigLookupFrequency	Time between config lookups, in milliseconds.
HomeStoreRefreshIntervalMins	Time between Home Store refreshes, in minutes.
HeartbeatServiceMode	Mode to use for the heartbeat service.

HeartbeatIntervalMins	Heartbeat send interval, in minutes.
SmtpServer	Address of SMTP server to use.
SmtpUser	Account name for SMTP account.
SmtpPassword	Password for SMTP account.
NotificationEmail	Email address to use for notifications.

Users.xml

Defines the users and groups for a study. The following example shows the contents of Users.xml for the "dummy" configuration:

```
<Group Name="Everyone">
  <User Name="systemlow" Password="" ActiveFrom="0001/1/1 12:00:00 AM"
ActiveUntil="2100/12/31 11:59:59 PM" />
  <User Name="systemhigh" Password="highpass" ActiveFrom="0001/1/1 12:00:00 AM"
ActiveUntil="2100/12/31 11:59:59 PM" />

<Group Name="Residents">
  <Group Name="Adults">
  </Group>

  <Group Name="Kids">
  </Group>
</Group>

<Group Name="Guests">
</Group>
</Group>
```