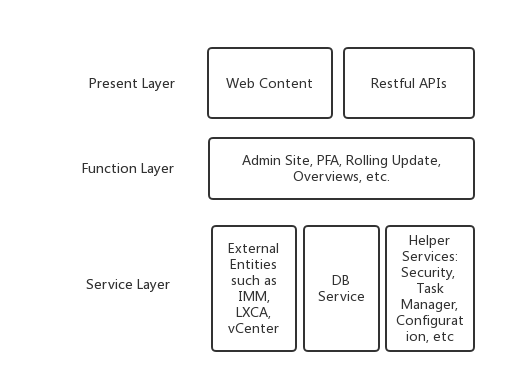
UnifiedService developer guide

The “UnifiedService” Project (we often call it UUS) is the backend service of LXCI for both VMware vCenter and Microsoft System Center. It is in where the implementation of most LXCI features locates. This project is mainly written in Python.

Logically, we can break down the UUS project into below view:



As in the past we didn’t follow any coding discipline, the source code is a bit messy that you may find weird code in weird place, i.e. the logic is not well isolated as you can see in the above logical view.

Below we list LXCI’s main components with brief description, code entry location and owner.

***Note:*** *the locations listed below are mainly the entry points, the complete implementation of a feature could spread over several packages which may not be listed below.*

[The Present Layer 2](#_Toc486596506)

[1. Web Content ( html, javascript, css, etc) 2](#_Toc486596507)

[a) Web content for the administration site of LXCI VMware 2](#_Toc486596508)

[b) Web contents that are shared with LXCI VMware, SCVMM and SCOM 2](#_Toc486596509)

[2. Rest APIs for web client 2](#_Toc486596510)

[a) APIs for the administration site of LXCI VMware 2](#_Toc486596511)

[b) RESTful APIs for LXCI main features. 3](#_Toc486596512)

[The Function Layer 3](#_Toc486596513)

[1. Administration Site (available in LXCI VMware virtual appliance edition only) 3](#_Toc486596514)

[2. Overview features 3](#_Toc486596515)

[3. Predictive Failure Alert (PFA) 4](#_Toc486596516)

[4. Rolling Update and Reboot 4](#_Toc486596517)

[5. Other host management features 4](#_Toc486596518)

[The Service Layer 4](#_Toc486596519)

[1. External Entities Interaction 4](#_Toc486596520)

[a) IMM/Host 4](#_Toc486596521)

[b) LXCA 5](#_Toc486596522)

[c) vCenter 5](#_Toc486596523)

[d) System Center 5](#_Toc486596524)

[2. Security 6](#_Toc486596525)

[3. Database 6](#_Toc486596526)

[4. LXCI Configuration 6](#_Toc486596527)

[5. Background Tasks 6](#_Toc486596528)

The Present Layer

1. Web Content ( html, javascript, css, etc)
   1. Web content for the administration site of LXCI VMware
      1. Location: *UnifiedService\unified\_service\backend\webroot\WebContent\ApplianceConfiguration*
      2. Description:

This folder contains the web files for administration site of LXCI VMware. The master page is ‘manager.html’.

**Note:** Administration site is available in LXCI VMware virtual appliance edition only

* 1. Web contents that are shared with LXCI VMware, SCVMM and SCOM
     1. Location:  
        *UnifiedService\unified\_service\backend\webroot\WebContent*
     2. Description:

In above location you can find most of the web content of LXCI features such as Rolling Update, PFA, LXCA Registration, BMC Discovery, Configuration Pattern, etc.

These web contents are shared with LXCI VMware, SCVMM and SCOM.

* + - 1. LXCI VMware

There are 2 parts of GUIs for our LXCI vCenter plugin. One is a set of Flex files bundled into the plugin (i.e. IVPUI.zip), and the other is web files that are dynamically pulled from UUS by vCenter. The web files in above location are for the latter one. These web files, mostly written with dojo, are embedded into vSphere web client via frameset.

* + - 1. SCOM  
         There is one page “Discover/Authenticate BMC” used in SCOM
      2. SCVMM

There are 2 parts of GUIs for our scvmm add-in. One is add-in ui framework used to embed UI to SCVMM, and the other is web files that are dynamically pulled from UUS. The web files in above location are for the latter one. These web files, mostly written with dojo.

1. Rest APIs for web client
   1. APIs for the administration site of LXCI VMware
      1. Location:

*UnifiedService\unified\_service\backend\webroot\bin\ApplianceConfiguration\ Management.py*

* + 1. Description:

The python above is the entry point of all RESTful APIs for administration site.  
**Note:** Administration site is available in LXCI VMware virtual appliance edition only

* 1. RESTful APIs for LXCI main features.

In below locations, you can find the RESTful APIs for most LXCI features such as Rolling Update, PFA, BMC Discovery, Configuration Pattern, etc. These APIs are shared with LXCI VMware, SCVMM and SCOM.

*UnifiedService\unified\_service\backend\webroot\bin\pages\*

*UnifiedService\unified\_service\backend\webroot\bin\rest\*

*UnifiedService\unified\_service\backend\webroot\bin\common\CommonAPI.py*

*UnifiedService\unified\_service\backend\webroot\bin\task\RestTask.py*

*UnifiedService\unified\_service\backend\webroot\bin\xHMC\RestxHMC.py*

*UnifiedService\unified\_service\backend\webroot\bin\ras\RASHistory.py*

*UnifiedService\unified\_service\backend\webroot\bin\vRealize\RestvRO.py*

**Tips:** *usually, when you want to find the python file of an API, you can check the source code in UnifiedService\unified\_service\backend\webroot\bin\ServerStart.py. It contains the code to initialize UUS RESTful API service and you can locate the corresponding file there.*

The Function Layer

In below we document the main functional components only. If you are a new comer, these components are good starting points for studying our product.

1. Administration Site (available in LXCI VMware virtual appliance edition only)
   1. Owner: **TBD**
   2. Location:
      1. *UnifiedService\unified\_service\backend\webroot\bin\ApplianceConfiguration\App\*.py*
      2. *UnifiedService\unified\_service\configuration\linux\script\\*.sh*
   3. Description:

This feature provides user a website to configure varies settings of LXCI virtual appliance, such as software upgrade, network access, data backup/restore, etc. The implementation of this feature can be found in above locations which contain pythons and shell scripts.

1. Overview features
   1. Owner: **TBD**
   2. Location:
      1. Cluster overview:  
         *UnifiedService\unified\_service\backend\webroot\bin\pages\ ClusterHostInformation.py*
      2. Host overview:

*UnifiedService\unified\_service\backend\webroot\bin\pages\IVP.py*  
*UnifiedService\unified\_service\backend\webroot\bin\platform\VMware\GetVCHostInfoService.py*

*UnifiedService\unified\_service\backend\webroot\bin\ cim\CIMInsCollection.py*

* 1. Description:

The most often use overviews are cluster overview and host overview, the former shows the number of hosts, host access states, request BMC/Host access, etc., the latter shows individual host states such as CPU, memory, firmware, etc.

The logic of overviews can be separated into 2 parts, one is background tasks to collect data from IMM and vCenter or System Center periodically, the other is packing and organizing the collected data and to send to present layer (via RESTful APIs).

1. Predictive Failure Alert (PFA)
   1. Owner: **TBD**
   2. Location

*UnifiedService\unified\_service\backend\webroot\bin\ras*

*UnifiedService\unified\_service\backend\webroot\bin\pfa*

*UnifiedService\unified\_service\backend\webroot\bin\rest\IMMCIMListener.py*

* 1. Description

The implementation of this feature is under the “RAS” and “PFA” package. The entry point is processIMMIndication() in “IMMCIMListener.py”. When an IMM indication is received the IMMCIMListener stores it into database and pass it to RASEngine for PFA handling. The RASEngine checks the policy for the outstanding indication and determines whether VMs in the host should be migrated. The policy is configured in vCenter or System Center.

1. Rolling Update and Reboot
   1. Owner: **TBD**
   2. Location

*UnifiedService\unified\_service\backend\webroot\bin\rsu*

*UnifiedService\unified\_service\backend\webroot\bin\pages\RollingUpdate.py*

* 1. Description

The “RSU” package contains all the implementation of rolling update. The entry point (RESTful API) is at “RollingUpdate.py”.

There are 2 types of rolling update: single and cluster. The former updates single host and the latter updates a group of machines. Update task can be configured in vCenter or System Center and executed through LXCA (update with policy) or OneCli (update without policy). Newly created tasks are firstly stored in database; a background thread which runs periodically reads the tasks from database and executes the updates accordingly.

1. Other host management features

There are other features like power capping, chassis map, etc. These features do not change very often so they are not documented here. Generally you can locate the source code of these features by locating the corresponding RESTful APIs.

The Service Layer

1. External Entities Interaction
   1. IMM/Host
      1. Owner: **TBD**
      2. Location

*UnifiedService\unified\_service\backend\webroot\bin\cim*

* + 1. Description

UUS interacts with IMM and host through the CIM package. UUS does this usually for collecting host/IMM information, subscribing IMM indication, etc.

* 1. LXCA
     1. Owner: **TBD**
     2. Location

*UnifiedService\unified\_service\backend\webroot\bin\xHMC*

* + 1. Description

UUS interacts with LXCA through the “xHMC” package. The RESTful API entry is in “RestxHMC.py”. Currently LXCA is used to implement features like configure pattern, chassis map and rolling update, etc.

* 1. vCenter
     1. Owner: **TBD**
     2. Location

*UnifiedService\unified\_service\backend\webroot\bin\platform\VMware*

*UnifiedService\unified\_service\backend\webroot\bin\platform\VMware\vSphereSDKClient.py*

*UnifiedService\unified\_service\backend\webroot\bin\jrestclient*

* + 1. Description

UUS talks to vCenter in scenarios likes registering vCenter plug-in, collecting host properties, pushing IMM indications to vCenter, etc.

The pythons under the “VMware” package contain logics of vCenter related functions, such as plug-in registration, VM migration, collecting host information, etc. Since UUS is written in python and VMware doesn’t provide python based vCenter SDK, we create 2 helper projects working as proxies to talk to vCenter with its java based SDK

In the old days, UUS talks to vCenter via the “ivp-tool.jar”. This is a subproject under “eh\_tvp” in GIT. UUS does this by invoking a java command line to execute a java class defined in “ivp-tool.jar”. The java class implements the logic to talk to vCenter.

We later create another project “jrestapi” (under “eh\_tvp” as well). It is going to be a replacement of the “ivp-tool”. It provides RESTful APIs to talk to vCenter. Instead of invoking java command line which reconstructs JVM instance and vCenter connection every time, “jrestapi” keeps a fix vCenter connection and JVM instance; this way save a lot of time on every vCenter talk.

The “vSphereSDKClient.py” wraps java command lines as python methods to talk to vCenter via “ivp-tool.jar”.

The “jrestclient” package contains the client code calling the RESTful APIs of “jrestapi”.

* 1. System Center
     1. Owner: **TBD**
     2. Location

*UnifiedService\unified\_service\backend\webroot\bin\platform\microsoft*

* + 1. Description  
       1. UUS communicates with SCOM in two main ways, 1) provide the server inventory data by rest api. 2) send the system event to SCOM when receive a CIM indication from managed IMM. For SCOM, we use the code “MicrosoftPlatform.py”, “SCOMSDKClient.py” and other powershell script under \platform\microsft folder.

2. UUS communicates with SCVMM by rest api and powershell. For SCVMM, we use the code “MicrosoftPlatform.py”, “SCVMMSDKClient.py and other powershell script under \platform\microsft folder.

1. Security
   1. Owner: **TBD**
   2. Location

*UnifiedService\uim\_crypt*

* 1. Description

In here the security we talk is all about encryption/decryption. UUS never stores plain text password or sensitive data into database. All passwords are hashed (HMAC\_SHA256) and sensitive data encrypted (AES-256 GCM) with openssl and dedicated keys. The keys are generated randomly at LXCI installation/deployment.

The above folder contains several C++ projects and a python package. The C++ projects wrap openssl library into command line tools/library for data encryption/decryption. The python package uses the command line tools to build higher level security logic.

1. Database
   1. Owner: **TBD**
   2. Location

*UnifiedService\unified\_service\backend\webroot\bin\model\database.py*

*UnifiedService\unified\_service\tools\win32\DBmigration\DBmigration\_Postgresql*

* 1. Description

The above python defines all tables for LXCI database.

Database table is accessed directly (i.e. no proxy class) across the whole UUS project. (This is a bad practice and should be changed!)

We use Alembic to manage our database scheme. To change the database scheme, you must (1) update/add the database table into “database.py”, (2) use alembic tool to generate database upgrade script in the folder “DBmigration\_Postgresql” listed above.

1. LXCI Configuration
   1. Location
      1. Configure file for LXCI VMware virtual appliance edition

*UnifiedService\unified\_service\configuration\uus\global.conf*

* + 1. Configure file for LXCI windows platforms (LXCI VMware, SCOM, SCVMM)

*UnifiedService\unified\_service\InstallShield\bin\global.conf*

* + 1. Configure file reader

*UnifiedService\unified\_service\backend\webroot\bin\config\ConfigReader.py*

* 1. Description

The global.conf contains all our LXCI settings, such as data collecting intervals, database name, etc. The “ConfigReader.py” should be used to access the configure items in the configure file.

1. Background Tasks

Several background tasks are initialized when UUS starts. Jobs like collecting IMM/host data, executing rolling update, etc are performed in these background tasks. You can check below pythons to see how they are initialized:

*UnifiedService\unified\_service\backend\webroot\bin\ServerStart.py*

*UnifiedService\unified\_service\backend\webroot\bin\StartMonitor.py*