

UBMS Web Environment

Build Manual

***August 2014***

***Revision 1.0.1***

***Acknowledgements***

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

A license is hereby granted to copy and reproduce this specification for internal use only.

No other license, express or implied, by estoppel or otherwise, to any other intellectual property rights is granted herein.

Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

This specification is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

\*Other names and brands may be claimed as the property of others.

Copyright © 2008 - 2014 Intel Corporation. All rights reserved.

***Revision History***

|  |  |  |
| --- | --- | --- |
| **Revision** | **Revision History** | **Date** |
| 1.0.0 | Initial release . | August 2014 |

***Contents***

***Tables***

Table 1.EDK Build Infrastructure Support Matrix ................................................................... 1

***Figures***

Figure 1. UEFI/PI Firmware Image Creation.................................................................... 11

**1**

**Introduction**

Necessary module:

UBMS

Python

Git

Pip

RabbitMQ

Flask==0.10.1

Flask-Bootstrap==3.1.1.2

Flask-Celery==2.4.3

Flask-Mail==0.9.0

Flask-Migrate==1.2.0

Flask-Moment==0.3.3

Flask-SQLAlchemy==1.0

Flask-Script==2.0.5

Flask-WTF==0.9.5

Jinja2==2.7.2

WTForms==1.0.5

Werkzeug==0.9.4

celery==3.1.13

### install python

### install pip

python setup.py install

### install virtualenv

pip install virtualenv

### install Flask

<https://pypi.python.org/pypi/Flask/0.10.1>

Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions. And before you ask: It's BSD licensed!

### install Erlang

<http://www.erlang.org/>

**Erlang** is a general-purpose [concurrent](http://en.wikipedia.org/wiki/Concurrent_computing), [garbage-collected](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) [programming language](http://en.wikipedia.org/wiki/Programming_language) and [runtime](http://en.wikipedia.org/wiki/Run_time_system" \o "Run time system)system. The sequential subset of Erlang is a [functional language](http://en.wikipedia.org/wiki/Functional_language), with [eager evaluation](http://en.wikipedia.org/wiki/Eager_evaluation), [single assignment](http://en.wikipedia.org/wiki/Single_assignment), and [dynamic typing](http://en.wikipedia.org/wiki/Dynamic_typing). It was designed by [Ericsson](http://en.wikipedia.org/wiki/Ericsson) to support distributed, [fault-tolerant](http://en.wikipedia.org/wiki/Fault-tolerance), [soft-real-time](http://en.wikipedia.org/wiki/Soft_real-time), non-stop applications. It supports[hot swapping](http://en.wikipedia.org/wiki/Hot_swapping), so that code can be changed without stopping a system.

### install RabbitMQ

<http://www.rabbitmq.com/>

**RabbitMQ** is open source [message broker](http://en.wikipedia.org/wiki/Message_broker) software (sometimes called [message-oriented middleware](http://en.wikipedia.org/wiki/Message-oriented_middleware)) that implements the [Advanced Message Queuing Protocol](http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol) (AMQP). The RabbitMQ server is written in the [Erlang programming language](http://en.wikipedia.org/wiki/Erlang_(programming_language)" \o "Erlang (programming language))and is built on the [Open Telecom Platform](http://en.wikipedia.org/wiki/Open_Telecom_Platform) framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages.

### install kombu

<https://pypi.python.org/pypi/kombu>

The aim of Kombu is to make messaging in Python as easy as possible by providing an idiomatic high-level interface for the AMQ protocol, and also provide proven and tested solutions to common messaging problems.

### install pika

<https://pypi.python.org/pypi/pika/0.9.14>

Pika is a pure-Python implementation of the AMQP 0-9-1 protocol that tries to stay fairly independent of the underlying network support library. Pika was developed primarily for use with RabbitMQ, but should also work with other AMQP 0-9-1 brokers.

### install flower

<https://pypi.python.org/pypi/flower/0.7.2>

Flower is a web based tool for monitoring and administrating Celery clusters.

### install librabbitmq

<https://pypi.python.org/pypi/librabbitmq>

Python bindings to the RabbitMQ C-library [rabbitmq-c](https://github.com/alanxz/rabbitmq-c). Supported by Kombu and Celery.

If you’re using RabbitMQ (AMQP) as the broker then you can install the librabbitmq module to use an optimized client written in C:

$ pip install librabbitmq

The ‘amqp’ transport will automatically use the librabbitmq module if it’s installed, or you can also specify the transport you want directly by using the pyamqp:// or librabbitmq:// prefixes.

### install redis

<https://pypi.python.org/pypi/redis/>

Python client for Redis key-value store