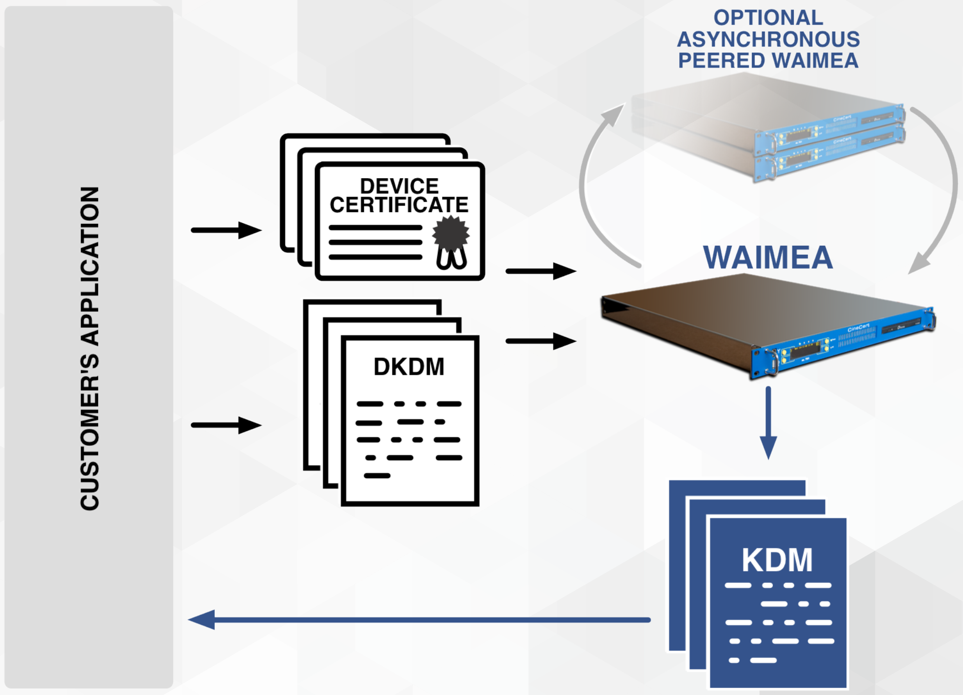
WAIMEA

CineCert

D-Cinema Key Management Server

VALIDATING KDM GENERATOR

- *Industry Standard Implementation*
- *Fast, Reliable KDM Generation*
- *Secure Processing of Content Keys*
- *High Availability Configurations Available*
- *Easily Programmable and Scriptable*
- *Robust Python, C++ and REST APIs*



CineCert's Waimea Key Management System is a secure hardware device for the creation of encryption keys and Key Delivery Messages (KDMs), and for the management of libraries of digital identity certificates. Waimea is designed to integrate seamlessly into your existing workflow by providing all functions using remote procedure call (RPC). System redundancy is enhanced by secure, automated peering with off-site Waimea servers.

CineCert, LLC 2840 North Lima Street, Suite 110A, Burbank, CA 91504 USA
PHONE (818) 563-1455 **FAX** (818) 563-1459 **E-MAIL** sales@cinecert.com **WEB** www.cinecert.com

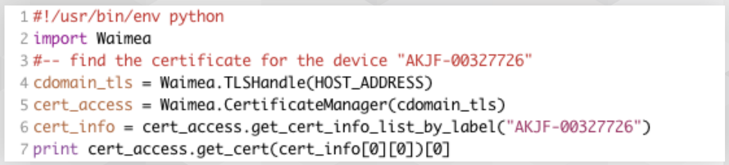
R E L I A B L E D I G I T A L C I N E M A

The Waimea D-Cinema Key Management Server is a mature, enterprise-class KDM processing solution. Top-tier distribution providers rely every day on Waimea for precise operations on d-cinema Key Delivery Messages. Detailed, personal support is provided by our knowledgable technical staff.

Waimea provides the following critical functions for mastering, authoring and distribution of Digital Cinema Packages (DCP):

Certificate Storage & Validation

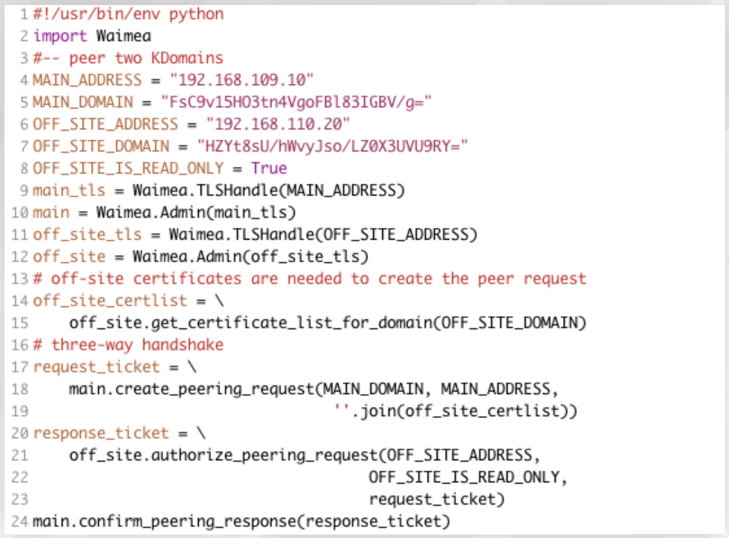
Waimea will ingest and validate a d-cinema device certificate using chain elements previously loaded onto the server via the same process. Once a certificate has been loaded, it will be available for KDM authoring operations. Waimea supports certificates in both SMPTE 430-2 and JPEG Interop formats.



```python
#!/usr/bin/env python
import Waimea
#-- find the certificate for the device "AKJF-00327726"
cdomain_tls = Waimea.TLSHandle(HOST_ADDRESS)
cert_access = Waimea.CertificateManager(cdomain_tls)
cert_info = cert_access.get_cert_info_list_by_label("AKJF-00327726")
print cert_access.get_cert(cert_info[0][0])[0]
```

DKDM Storage & Validation

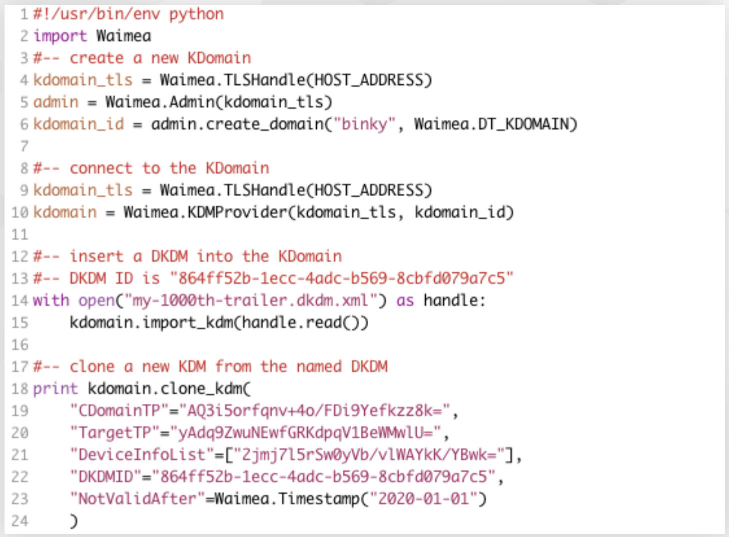
Waimea stores Distribution KDMs (DKDMs) for future use in KDM authoring. DKDMs are analyzed for errors during ingest. Waimea limits KDM authoring to the time limits present in the source DKDM.



```python
#!/usr/bin/env python
import Waimea
#-- peer two KDomains
MAIN_ADDRESS = "192.168.109.10"
MAIN_DOMAIN = "FsC9v15HO3tn4VgoFBl83IGBV/g="
OFF_SITE_ADDRESS = "192.168.110.20"
OFF_SITE_DOMAIN = "HZYt8sU/hWvyJso/LZ0X3UVU9RY="
OFF_SITE_IS_READ_ONLY = True
main_tls = Waimea.TLSHandle(MAIN_ADDRESS)
main = Waimea.Admin(main_tls)
off_site_tls = Waimea.TLSHandle(OFF_SITE_ADDRESS)
off_site = Waimea.Admin(off_site_tls)
# off-site certificates are needed to create the peer request
off_site_certlist = \
    off_site.get_certificate_list_for_domain(OFF_SITE_DOMAIN)
# three-way handshake
request_ticket = \
    main.create_peering_request(MAIN_DOMAIN, MAIN_ADDRESS,
                                 ''.join(off_site_certlist))
response_ticket = \
    off_site.authorize_peering_request(OFF_SITE_ADDRESS,
                                       OFF_SITE_IS_READ_ONLY,
                                       request_ticket)
main.confirm_peering_response(response_ticket)
```

KDM Generation

Using DKDM and a target certificate already ingested, Waimea will create a KDM for the target device. Waimea allows both single-source DKDM with feed-through of DKDM metadata such as CPL identity and forensic marking flags ("clone" mode) and also multiple source DKDMs with a source CPL to be used to produce a KDM with a composite set of keys and customized metadata (Create mode). For users supporting legacy devices, Waimea automatically selects the correct KDM format (SMPTE or JPEG Interop) based on the target certificate format.



```python
#!/usr/bin/env python
import Waimea
#-- create a new KDomain
kdomain_tls = Waimea.TLSHandle(HOST_ADDRESS)
admin = Waimea.Admin(kdomain_tls)
kdomain_id = admin.create_domain("binky", Waimea.DT_KDOMAIN)

#-- connect to the KDomain
kdomain_tls = Waimea.TLSHandle(HOST_ADDRESS)
kdomain = Waimea.KDMProvider(kdomain_tls, kdomain_id)

#-- insert a DKDM into the KDomain
#-- DKDM ID is "864ff52b-1ecc-4adc-b569-8cbfd079a7c5"
with open("my-1000th-trailer.dkdm.xml") as handle:
    kdomain.import_kdm(handle.read())

#-- clone a new KDM from the named DKDM
print kdomain.clone_kdm(
    "CDomainTP"="AQ3i5orfqnv+4o/FDi9Yefkzz8k=",
    "TargetTP"="yAdq9ZwuNEwfGRKdpqV1BeWMwlU=",
    "DeviceInfoList"=["2jmj7l5rSw0yVb/vlWAYkK/YBwk="],
    "DKDMID"="864ff52b-1ecc-4adc-b569-8cbfd079a7c5",
    "NotValidAfter"=Waimea.Timestamp("2020-01-01")
    )
```

Client Libraries

Waimea is supplied with client libraries to allow customers' applications to request services from the server. Supported API languages and platforms include:

- C++ source code (*win32/Unix*)
- Python (*win32, Unix*) • REST (*any HTTPS client*)

Box Contents

The Waimea server package includes the following:

- The Waimea Server
- A/C Power Cable
  (NEMA 5-15 plug type / IEC C13 connector and/or CEE 7/7 plug type / IEC C13 connector)

The Waimea client SDK software is available for a variety of platforms.

Waimea Specifications

The Waimea server dimensions are 21"L x 19"W x 1.75"H (1U high). It is 22" deep including the front handles. The server has an auto-switching power supply and can be connected to input voltages of 120v or 240v, 50 or 60Hz. The Waimea server uses approximately 50 watts.

The server can either sit on a rack shelf or be mounted in the rack by the front-mounted rack ears. The holes on the rack ears are not threaded.