BlackBerry Dynamics User Certificate Management Protocol

Version 1.2b

1 Overview

The BlackBerry Dynamics platform supports enrolling and downloading users' enterprise-issued certificates and their use in the BlackBerry Dynamics runtime for various purposes, such as S/MIME and client-certificate-based authentication for TLS connections.

This document specifies the protocol used by the Good Control server and BlackBerry UEM server to fetch/enroll user certificates. An enterprise backend server that implements this protocol is called a PKI Connector and is implemented by the customer. This protocol runs over HTTPS and defines JSON-formatted messages.

The Good Control server and BlackBerry UEM server will use the APIs mentioned in this document when BlackBerry Dynamics runtime makes a request for a user's enterprise certificate. Features and protocols documented here are used by the Good Control server version 2.1 and above or BlackBerry UEM server version 12.7 or later and are available to apps built using BlackBerry Dynamics SDK 2.1 and above.

Customers must build their own PKI Connector that implements this protocol and interfaces with their enterprise Certificate Authority (CA). An example implementation of such a PKI Connector is described at PKI Cert Creation via Good Control: Reference Implementation.

1.1 Background

This document assumes you are aware of features and architecture of the BlackBerry Dynamics Platform. For background, you might want to review these introductory articles on the BlackBerry Developer Network.

- BlackBerry Dynamics Administrator and Developer Overview

Terminology used in this document:

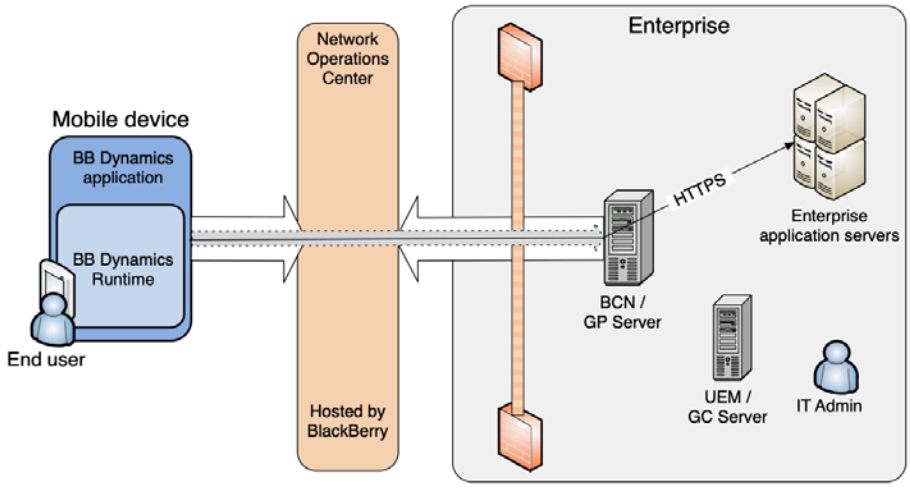
BlackBerry Dynamics App: An app with embedded calls to the BlackBerry Dynamics runtime which provides services/features to the user. When we refer to a BlackBerry Dynamics app we mean the application layer logic.

BlackBerry Dynamics Runtime: Every BlackBerry Dynamics app includes an instance of the BlackBerry Dynamics runtime. The runtime has an API that gives the app access to user authentication, secure communications, secure storage, and communication behind the firewall. The runtime also handles enforcement of security policies on behalf of the application. An instance of the BlackBerry Dynamics runtime may sometimes be referred to as a *Container*.

GC Server: Good Control (GC) server typically installed on premise manages users, devices, policies, BlackBerry Dynamics applications. After activation the BlackBerry Dynamics runtime connects with the GC server to fetch its policies, actions, user certificates etc. An administrator uses the Good Control console hosted by GC server to perform these management functions.

BlackBerry UEM Server: Customers can also use BlackBerry UEM server to activate BlackBerry Dynamics applications. BlackBerry UEM server provides similar functionality as the Good Control server in addition to supporting MDM, native applications on various mobile platforms. Additional information about BlackBerry UEM can be found here. All references to Good Control server in this document applies equally to BlackBerry UEM server.

1.2 Components



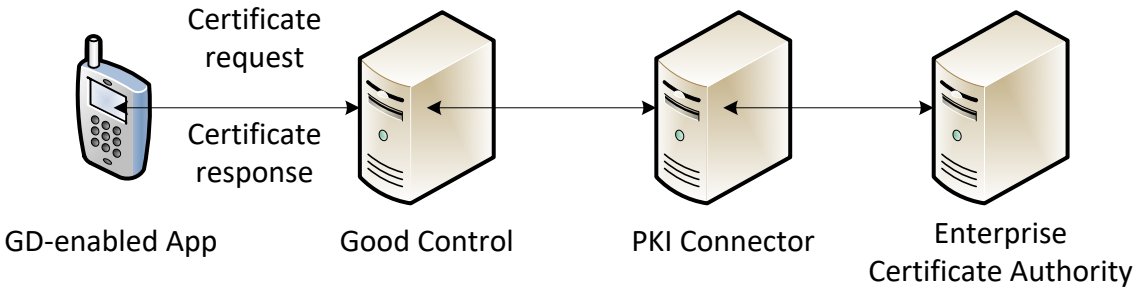
2 User Certificate

The BlackBerry Dynamics platform supports enrolling and downloading users' enterprise-issued certificates and their use in the BlackBerry Dynamics runtime for various purposes, such as S/MIME and client-certificate-based authentication for TLS connections.

The GC server supports:

1. Pushing certificates to the BlackBerry Dynamics runtime when uploaded in PKCS12 format to the GC server (manual mode). User certificates can be uploaded by the user (using self-service console), by the administrator, or by an enterprise's tool using the GC's SOAP APIs.
2. Automatic fetching of users' enterprise certificates from the PKI Connector (on-demand mode).

2.1 High Level Overview



2.2 Settings in Good Control

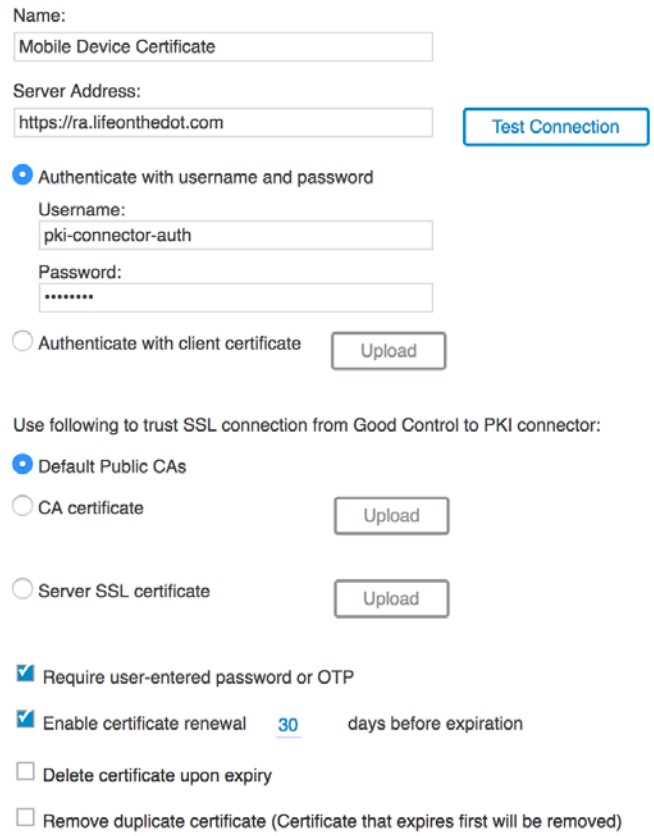
In Good Control server Certificate Definitions allow Dynamics applications to obtain user certificates from your organization's PKI Connector. Administrator needs to do following actions for BlackBerry Dynamics apps to get user certificates.

1. Enable "Allow use of client certificates" setting in Security Policies in the Policy Set assigned to the user.
2. Create Certificate Definition and provide PKI Connector information
3. Allow applications to receive certificate.

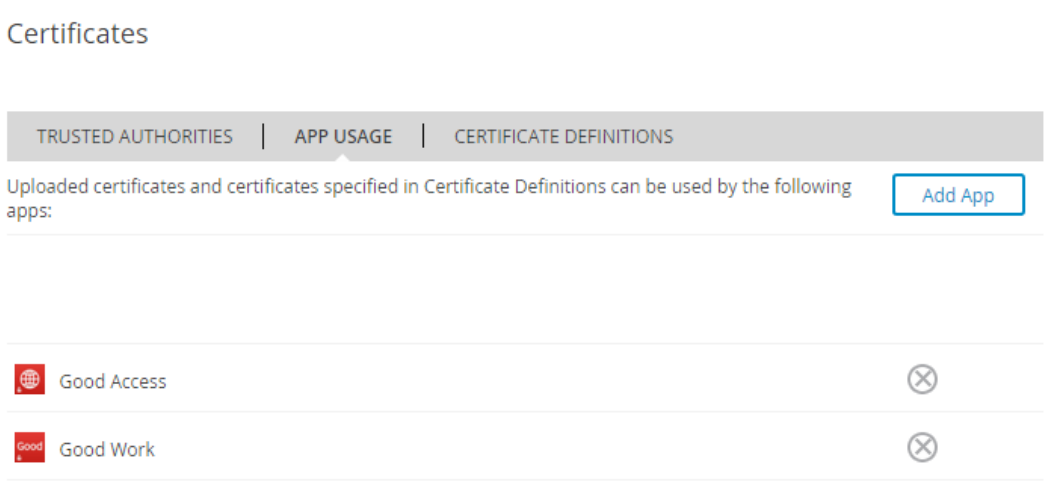
Administrators can create certificate definitions in the GC console. A certificate definition has a name, URL for the PKI Connector, and some attributes for the certificate such as OTP/password requirement and so on (see the screenshot below). Some of these attributes are sent to the BlackBerry Dynamics runtime of the BlackBerry Dynamics apps that are allowed to have client certificates. The Good Control server will connect to the PKI Connector specified here.

For the applications that are allowed access to the user certificate, user must enroll the user certificate before they can use an app.

Certificate Definition screen in the GC console



GC screen to specify the applications allowed to fetch the user certificate



2.3 Settings in BlackBerry UEM

In BlackBerry UEM server User credential profiles allow Dynamics applications to obtain user certificates from your organization's PKI Connector. User Credential Profile needs to be assigned to the user. Additional information about User credential profile can be found at these links.

- User Credential Profile
- **C**reating a User Credential Profile to connect to your PKI Connector

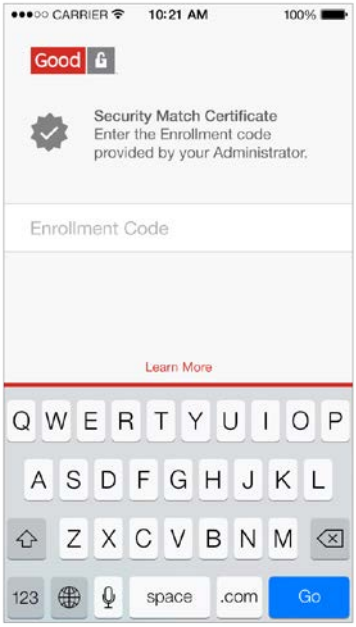
2.4 Certificate Enrollment

When "User Certificate" feature is enabled for the user, BlackBerry Dynamics runtime on activation receives the Certificate Definition. BlackBerry Dynamics runtime will complete enterprise certificate enrollment before the application can be used by the user. Based on the policy BlackBerry Dynamics runtime may prompt the user for the enrollment code.

BlackBerry Dynamics runtime sends the enrollment request to the GC server and receives user key-pair in the pkcs-12 format as shown below.

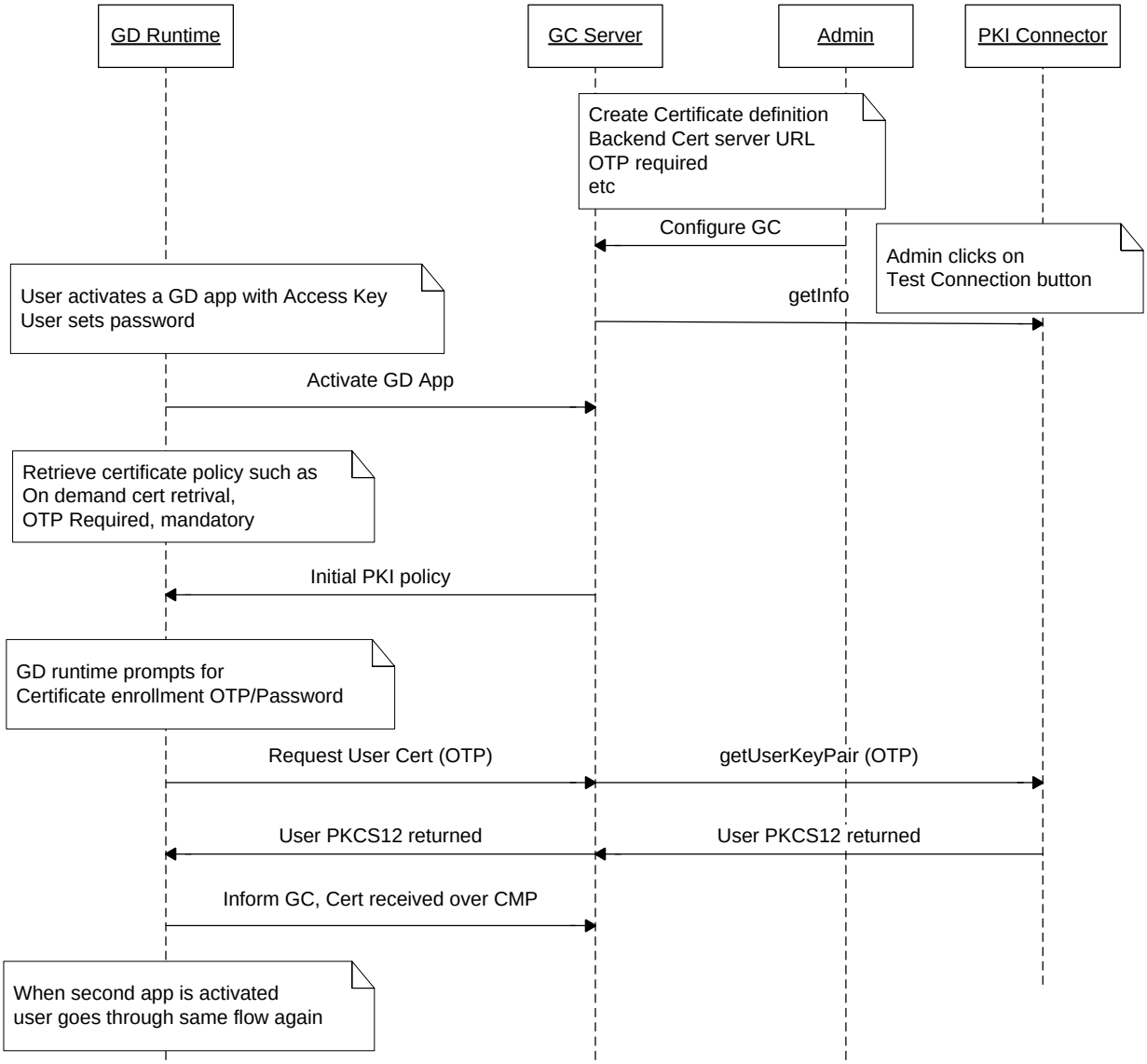
2.4.1 User Input in BlackBerry Dynamics Runtime

Based on the settings on the server, user may be prompted to enter password/OTP in the Dynamics application.



2.4.2 Certificate Enrollment Flow

Flow diagram below illustrates sequence of events that take place between BlackBerry Dynamics runtime, GC server, and the PKI Connector to fetch user certificate.



2.5 Certificate Renewal

2.5.1 Renew User Experience

Certificate is renewed by the BlackBerry Dynamics runtime at the configured time window set in the console. User is not provided any UI when certificate is renewed successfully.

At the following triggers, certificate is checked for the renewal

- Unlocking after application cold startup
- Idle unlock
- Unlocking when restoring from the background

Once one app has renewed the certificate, all other applications on the device are informed about the presence of the new certificate by the GC server. All other applications will fetch the renewed certificate locally when they are next started up or unlocked. Auth delegate app is given preference and enters renew time window three days earlier than other apps.

Admin can force renew for a user certificate or for all users via manual action from the console.

2.5.2 Handling Error Conditions

- On temporary error, BlackBerry Dynamics runtime will try to renew again at every hour. (such as PKI connector or GC server not reachable)
- On unknown and unexpected again, BlackBerry Dynamics runtime will try again in twenty-four hours.
- Renew attempt will be stopped when user certificate expires. No UI is provided to the user.
    - User is informed about the expired certificate when they unlock the application. User is then prompted with new certificate enrollment flow.

2.5.3 Renew Protocol Flow

1. BlackBerry Dynamics runtime sends renewal request to the PKI Connector. This request is a cms SignedData object, signed with user's current private key containing a pkcs10 payload (see the format in the getUserKeyPair2 section).

2. PKI Connector must implement following interface
    a. getUserKeypair2 interface to return a P12 containing new key-pair and public certificate for the user. In this scenario BlackBerry Dynamics runtime will import the new key-pair sent the PKI Connector.
3. PKI connector is sent an acknowledgement, after BlackBerry Dynamics runtime has received the renewed certificate.

2.6 Certificate removal

When a user is removed or a device is removed from Good Control server or the certificate is deleted by the admin or user, Good Control server will notify the PKI Connector, about the user's public certificate that is no longer used in the BlackBerry Dynamics deployment. The PKI Connector can revoke these certificates.

3 PKI Connector interactions

GC server and BlackBerry UEM server make the following API calls.

3.1 Authentication

GC server will support following authentication schemes as requested by the backend server. These authentication methods can be HTTPS basic auth or SSL client authentication.

3.2 Interfaces

1. getInfo
2. getUserKeyPair2 (initial enrollment and renew)
3. notifyCertificateReceived (optional)
4. notifyCertificateRemoved (optional)
5. getUserKeyPair (deprecated, only initial enrollment is supported)

3.3 GetInfo API

This API is called to detect all the commands the PKI Connector has implemented. This command is also used to verify the authentication credentials provided in the GC console by the administrator. Test Connection feature in the GC console uses this command.

If this command is not implemented GC will assume only getUserKeyPair is supported by the PKI Connector.

HTTP request-line:
GET customerSpecifiedPrefix/pki?operation=getInfo

customerSpecifiedPrefix optional. This is needed to specify where the service is hosted on the server when not hosted in the default path. It can be provided on the GC console.

Following JSON formatted response is expected in the HTTP body. Response contains all the commands implemented by the connector.

Element/Key	Type	Required	Comments
operations	Array of Strings	Y	Array of all the commands implemented by the PKI Connector

Auth failure: HTTP status code 401 is sent

Success: HTTP status code 200 with body mentioned above.

3.3.1 Sample request/response

Assuming on the GC console PKI Connector URL is set as
`https://ra.lifeonthedot.com`

```
GET /pki?operation=getInfo HTTP/1.0
Host: ra.lifeonthedot.com
```

```
Content-Type: application/json
Content-Length: 0

Response

HTTP/1.0 200 OK
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{
      “operations” : [“getInfo”, “getUserKeyPair2”]
}
```

3.4 getUserKeyPair2

This interface must support both fetching initial key-pair and renewing an existing certificate user has.

Initial Cert: When BlackBerry Dynamics apps is fetching the key-pair for the first time, GC will send mType value of “initialCert”. “authToken” will be sent if required by the admin. “reqId”, “deviceId”, “deviceName” are sent as they were sent in the “**getUserKeyPair2**” interface.

Renew Cert: When BlackBerry Dynamics app is renewing the certificate it already has, it uses its current private key to authenticate request to renew the certificate. This information is sent inside cmsSigned element by the BlackBerry Dynamics app.

HTTP request-line:
POST customerSpecifiedPrefix/pki?operation=getUserKeyPair2

Element/Key	Type	Required	Comments
mType	String	Y	{"initialCert", “renewCert”}
user	String	Y	User email addr or some other identifier. Subject for the certificate is created by the issuer.
authToken	String	N	OTP/password (for intialCert)
reqId	String	N	Only for initialCert To assist sender to match response

deviceId	String	N	Only for initialCert BlackBerry Dynamics device id. In some edge cases it is possible for BlackBerry Dynamics container to change its deviceId. This is here for informational and audit purposes. It could also be used by issuer to track if two apps on the same device are making request for the certificate at the same time (or very close in time).
deviceName	String	N	Only for initialCert Name of the device assigned by the user, which user can change at a later time. This is here for informational and audit purposes.
cmsSigned	Base64 encoded request	N	GC server sends this to the PKI Connector as received without doing any changes. See below

cmsSigned = base64 encoded (cms-signed-data(CertRequest))

CertRequest:

reqId	String	Y	To assist sender to match response
deviceId	String	N	BlackBerry Dynamics device id. In some edge cases it is possible for BlackBerry Dynamics app to change its deviceId. This is here for informational and audit purposes. It could also be used by issuer to track if two apps on the same device are making request for the certificate at the same time (or very close in time).
deviceName	String	N	Name of the device assigned by the user, which user can change at a later time. This is here for informational and audit purposes.
pkcs10	Base64	Y	Based64 encoded CSR

Response

Element/Key	Type	Required	Comments
status	String	Y	{success, failure}
failureInfo	String	N	
payloadType	String	N	=pkcs12

payload	Base64 encoded Object	N	Base64 encoded p12
reqID	String	Y	
password	String	N	If pkcs12 was password encrypted and authToken was not used for encryption, decryption password may be returned here. For Renew case password is needed

3.4.1 Initial Enrollment Sample

Assuming on the GC console PKI Connector URL is set as: *https://ra.lifeonthedot.com*

Request: Over the SSL connection to server htts://ra.lifeonthedot.com the following payload will be sent.

```
POST /pki?operation=getUserKeyPair2 HTTP/1.0
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{
      "mType": "initialCert",
      "user":   "joe.foo@lifeonthedot.com",
      "authToken": "56ht12d0",
      "reqId":  "12487",
       “deviceId”:  “6e8S8JCLN7Hc5v3cGqvfkfM/C/tAFDS1CFUPJ53ASL”,
      “deviceName”:  “Joe’s iPhone6”

}
```

If server URL was set as https://ra.lifeonthedot.com/foo in the GC console, the request will look like:

```
POST /foo/pki?operation=getUserKeyPair2 HTTP/1.0
Host: ra.lifeonthedot.com

Content-Type: application/json
Content-Length: XYZ
```

Response:

```
HTTP/1.0 200 OK
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{
```

```
	"status":"success",
	"reqId": "12487",
	"payloadType":"pkcs12",
	"password":"clearTextPassword",
	"payload":"BASE64 Encoded PKCS12"
}
```

3.4.2 Renew Sample

Request:

```
POST /pki?operation=getUserKeyPair2 HTTP/1.0
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{
	"mType": "renewCert",
	"user":   "joe.foo@lifeonthedot.com",
	"cmsSigned": "base64-enoded-CMS-signed-data"

}
```

Response:

```
HTTP/1.0 200 OK
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{

	"status":"success",
	"reqId": "12487",
	"payloadType":"pkcs12",
	"password":"clearTextPassword",
	"payload":"BASE64 Encoded PKCS12"
}
```

3.5 notifyCertificateReceived

When GC server receives confirmation that client has successfully imported the certificate, GC server will call notifyCertificateReceived on the PKI connector. PKI Connector may want this notification to take some action for the certificates that don't get delivered to the clients due to error in delivery. In addition PKI Connector, can trigger the Good Control server to remove old certificates for the user by returning list of X509s to be removed in the response.

This call will be invoked until connector responds with HTTP 200 and status value "success". In addition PKI connector can cause the Good Control server to try again by sending HTTP code 200, with status as failure, and failureInfo as "retry".

HTTP request-line:

POST customerSpecifiedPrefix/pki?operation=notifyCertificateReceived

Element/Key	Type	Required	Comments
user	String	Y	User email addr string
receivedCert	Base64	Y	Certificate received. DER encoded X509
otherCerts	array of Base64 encoded object	N	List of other certs present on the device/app ie. previous certs
deviceId	String	N	
deviceName	String	N	

Response:

Once HTTP response code of 200 is received, GC server will remove this notification task from the queue. For any other error code, GC will retry notification.

Element/Key	Type	Required	Comments
status	String	Y	{success, failure}
failureInfo	String	N	"retry" For any other failure code, notification will be stopped Such as "badReqeust", "unknownUser" etc
removeCerts	Array of Base64 encoded Object	N	DER encoded X509 GC will delete these certs

3.5.1 Sample

Request:

```
POST /pki?operation=notifyCertificateReceived HTTP/1.0
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{
      "user":   "joe.foo@lifeonthedot.com",
       "receivedCert": "base64-encode-x509"

}
```

Response:

```
HTTP/1.0 200 OK
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ
```

```
{
	"status":"success",
}
```

3.6 notifyCertificateRemoved

When a certificate is no longer in use by the BlackBerry Dynamics deployment, GC server will notify the PKI connector. This call will be invoked until connector responds with HTTP 200 and status value "success". If PKI connector returns failureInfo "retry", GC server will try again.

HTTP request-line:
customerSpecifiedPrefix/pki?operation=notifyCertificateRemoved

Element/Key	Type	Required	Comments
user	String	Y	User email addr
removedCerts	Array of Base64 String	Y	DER encoded X509
reason	String	N	“userRemoved”, ”certRemoved”, ”appRemoved”, ”duplicate”
deviceId	String	N	
deviceName	String	N	

Response:

Once HTTP response code of 200 is received, GC will remove this notification task from the queue. For any other error code, GC will retry notification.

Element/Key	Type	Required	Comments
status	String	Y	{success, failure}
failureInfo	String	N	“retry”

3.6.1 Sample

Request:

```
POST /pki?operation=notifyCertificateRemoved HTTP/1.0
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ
```

```
{
       "user":   "joe.foo@lifeonthedot.com",
       "removedCerts": ["base64-encode-x509"],
       "reason": "certRemoved"
}
```

Response:

```
HTTP/1.0 200 OK
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{

       "status":"success",
}
```

3.7 getUserKeyPair

This API is used to fetch a user certificate when key-pair has already been created by the backend certificate server. This request may be used for initial certificate request. Renew request will not be send on this interface.

Since there is new version of this API, you should start using getUserKeyPair2 instead of this one. If getUserKeyPair2 is supported, this API will not be called.

HTTP request-line:
customerSpecifiedPrefix/pki?operation=getUserKeyPair

customerSpecifiedPrefix is optional. This is needed to specify where the service is hosted on the server when not hosted in the default path. It can be provided in the GC console.

JSON formatted input sent in the HTTP body is as follows:

Element/Key	Type	Required	Comments
mType	String	Y	{"initialCert"}
user	String	Y	User email address or some other identifier. Subject for the certificate is created by the issuer.
authToken	String	N	OTP/password (for intialCert)
reqId	String	Y	To assist sender to match response
deviceId	String	N	BlackBerry Dynamics device id. This is here for informational and audit purposes. It could also be used by issuer to track if two apps on the same device are making request for the certificate at the same time (or very close in time). In some edge cases it is possible for BlackBerry Dynamics container to change its deviceId at a later time.

deviceName	String	N	Name of the device assigned by the user, which user can change at a later time. This is here for informational and audit purposes.

JSON formatted Response in the HTTP body. Response is a PKCS12 payload which may be encrypted.

Element/Key	Type	Required	Comments
status	String	Y	{success, failure}
failureInfo	string	N	See “Failure Reason”, below.
payloadType	String	N	=pkcs12
payload	Base64 encoded	N	pkcs12 containing user's private key and public certificate. It may or may not be encrypted. This will not be saved in the GC DB.
password	String	N	If the encryption password is same as the OTP provided by the user, no need to provide password. If pkcs12 was password encrypted and OTP was not used was encryption, decryption password may be returned here.
reqId	String	Y	reqID received in the request

3.7.1 Sample request/response

Assuming on the GC console PKI Connector URL is set as:
`https://ra.lifeonthedot.com`

Request: Over the SSL connection to server ra.lifeonthedot.com the following payload will be sent.

```
POST /pki?operation=getUserKeyPair HTTP/1.0
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{
      "mType": "initialCert",
      "user":   "joe.foo@lifeonthedot.com",
      "authToken": "56ht12d0",
      "reqId":  "12487",
       “deviceId”:  “6e8S8JCLN7Hc5v3cGqvfkfM/C/tAFDS1CFUPJ53ASL”,
      “deviceName”:  “Joe’s iPhone6”

}
```

If server URL was set as https://ra.lifeonthedot.com/foo in the GC console, the request will look like:

```
POST /foo/pki?operation=getUserKeyPair HTTP/1.0
Host: ra.lifeonthedot.com
```

```
Content-Type: application/json
Content-Length: XYZ

Response:

HTTP/1.0 200 OK
Host: ra.lifeonthedot.com
Content-Type: application/json
Content-Length: XYZ

{

      "status":"success",
      "reqId": "12487",
      "payloadType":"pkcs12",
      "password":"clearTextPassword",
      "payload":"BASE64 Encoded PKCS12"

}
```

3.8 Failure Reason

These errors may be returned by the PKI Connector.

Failure	Description
unknownUser	User does not exist or is not allowed
badRequest	Badly formatted request
unknownRequest	Requested action is not supported.
authFailure	Expired/incorrect OTP/password
badAlg	Unsupported/Unrecognized algorithm used
unknownCert	Certificate used/referenced in the operation not found
badMessageCheck	Signature/Integrity check failed
badTime	Time in the signature was not close enough
unknown	Any other errors will be treated as unknown error.