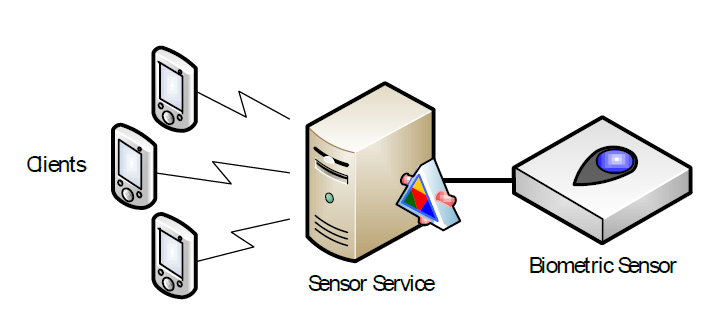
**CBP CEE Biometric Excellence Review of the OASIS WS-Biometric API Draft 01**

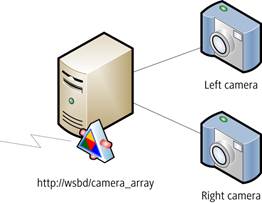
Engineers from OIT have reviewed the OASIS WS-Biometric API. The general approach to creating a service as the interface for devices was implemented here at CBP a few years ago and we were successful with that. We found some areas of concern and have a few questions.

1. Web Service Deployment – if a client is going to access an OASIS service on another IP address, there are latency issues to be concerned about that will interfere with the responsiveness of the device interaction to the client. If the diagram below indicates a connection through WIFI, the clients will be experiencing latency slowdown on their requests.



1. Stateless HTTPS Operation – there are concerns about going through a stateless connection from the client to the OASIS service. There is a challenge in this approach to know when the Client or OASIS service goes down and how to properly notify and clean up. We have implemented stateful socket connected local services to provide an interface for clients and devices here at CBP which has simplified and resolved this problem.
2. Binary Data is encoded – I am not sure why SOAP/w attachments weren’t specified as part of the communications standard. Reliance on encoding/decoding the binary data in the HTTP messages is very expensive, especially for mobile clients and when performing preview analysis imaging. We had tried this approach here a few years ago and we found it to be not sufficiently responsive to the client for preview analysis.
3. Orphaned Captured Images – captured images are cached on the service. The service notifies a client that a capture it has requested has been captured, then the client may request to down load it from the service. If the client neglects to download this image, how does this image get cleared? What if the client should die or exit without cleanup?
4. Thresholds for Preview Analysis – you may setup threshold parameters as part of the “set configuration” command. However the API doesn’t normalize the thresholds into standard values. So the normalization of these values will have to be handled on the client as each device may handle the threshold differently.
5. Preview Analysis – the client may specify that it wants preview analysis done by implementing the data pair of “livePreview” as part of the reader configuration. The main purpose of “livePreview” is to allow the client to get preview images and perform analysis. However, it’s not clear how “livePreview” is initiated nor how the streaming of images is stopped. If the OASIS service performs the capture on its own after returning preview images, then the service encapsulates the threshold while performing the preview analysis. If the service does not trigger a capture, then somehow the client has to initiate a capture.
6. The documentation states that “A client releases the lock upon completion of its desired sequence of tasks…. When lock ownership is transferred during a sensor operation, overlapping sensor operations are prevented by sensor operations returning sensorBusy”. Is the request to steal the lock queued or does the client have to submit another request to steal the lock?

1. Do we have any instances of implementation of this architecture?
2. If we do, then how well are the error recovery performed on behalf of a transaction when a component such as application or of the service abnormally ends?
3. Lock coordination and management. For example, how do we manage the case where due to a performance issue on the application, the lock appears to be stealable and consequentially it got stolen from another request?
4. How does a “livePreview” start and what does is the trigger(s) for it to stop and to signify the capture is completed (which is followed by a download request)?
5. A dataflow diagram would be helpful.
6. The ‘resources’ described are generally operations, and not resources. The interface could be considerably simplified by creating a set of defined resources along with the standard REST operations (GET, PUT, POST, DELETE, and we also would recommend supporting HEAD and OPTIONS).
7. Many of the operations would be better represented as standard operations sub-resources. For example, instead of the ‘try lock’ and ‘unlock’ separate operations, shown as /lock/{sessionId} with POST and DELETE, respectively, the URL would be something like /device/{deviceId}/lock, and the POST would carry a value of TRUE or FALSE. Alternatively, these could be viewed as an attempt to create a lock on a device, in which case the DELETE makes some sense.
8. Based on the above input, ‘errors’ can be dealt with in a couple of different ways. For genuine errors in the REST sense, we would strongly prefer a standard REST interface using HTTP error codes as appropriate – for example, we should get a 201 in response to a create through a PUT or POST only if the resource was created. This simplifies processing using standard HTTP client tools, including browsers.
9. When talking to many vendors, JSON implementation would probably be preferred by many implementers.
10. The base64encoding is unnecessary. This is not an inherent limitation of HTTP but relates to the attempt to pass mixed content in a single response, which there are better ways of dealing with in REST.
11. In this diagram, a service can control multiple cameras, can client A uses the left camera and client B uses the right camera simultaneously? Or the locking operation has control over the two cameras and only one application (one session id) has exclusive access to the two cameras.



1. In this diagram, what operation will allow the client application to select the Left or Right camera?
2. The “download” operation allows the transfer of the captured biometric data to the client.  What happens when the client does not call download? Is there a timer associated with each download to do the cleanup when the client fails to call the download operation?
3. Is there an operation to find out which client application has the lock and how long it has had it locked?
4. How does the client application controls the LED status light indicator of the device? Is there an operation command to controls the LED indicator.  For example, the 10-Print scanners has left hand, right hand LED indicator and other status light indicators to allow the client to turn on and off.
5. What API operation will allow client application reset or restart the device?
6. What happens when a client sends a request and does not get a result back? Is it recommended that the client implement a timeout for it’s requests?
7. Can a client restart the OASIS service?
8. How does OASIS resolve an unresponsive device? And who is responsible for restarting the service?
9. When an OASIS service is shutting down does it notify the clients? With a stateful service this is much easier when you can broadcast the event.