# NativeSPAssertionExport

The primary channel for communication of user information between the SP software and the web applications running behind it is the use of CGI request headers or environment variables. Some applications have advanced requirements for which access to actual SAML assertions can be useful. For these cases, the SP is able to provide the assertions it receives, but cannot do so in the same way due to size. Instead, a simple query mechanism is used.

If you don't know why you would need this feature, then you don't need it. Leave the default settings alone and ignore them.

In order for the export to be functional, the `exportLocation` and `exportACL` properties must be set for the relevant application's `<Sessions>` element. In addition, the `<SessionCache>` element must either omit, or turn on the `cacheAssertions` property. It defaults on, but may be explicitly turned off to optimize the cache when assertions don't need to be accessed later.

Then, when instructed to do so for a request (via the `exportAssertion` content setting), the application will be given a header or variable called `Shib-Assertion-Count` with the number of assertions that are available.

The URL to query for each assertion is passed in an individual header or variable named `Shib-Assertion-NN`, where `NN` is the two-digit sequence number of the assertion(`01`, `02`, etc). Performing a GET on that location will result in the assertion, with a MIME type of "application/samlassertion+xml".

Since the primary use case for this feature involves the forwarding or delegation of signed assertions, the XML is passed along unmodified from the issuer. As a result, although some security checking has been performed prior to caching, the data itself is left alone. Attribute filtering is **not** reflected in the results.

## Distributed Scenarios

In some cases, it's possible for the code that needs to perform the retrieval to be running on a different host from the SP itself. One such scenario would be a Java container fronted by Apache, such that Apache and the SP run separately from the Java container. The SP doesn't provide any real way of "securing" the retrieval step.

In particular, a non-localhost request would be subject to various attacks unless the retrieving code included some kind of TLS-based authentication check. The SP itself relies only on IP address checking to limit who can access the assertions, and this is much less safe when values other than the loopback address are allowed.

One solution to this problem is to use an SSH tunnel between the retrieving host and the host running the SP. An untested command to achieve this might look like:

```
ssh <username>@<sp.example.org> -L 80:localhost:80 -f -N
```