Base Language - Feature #6373

direct Java object access to converted Java code from in-JVM non-converted Java code

05/17/2022 10:34 AM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Feature #4065: server-side processing of client pl...	**WIP**
Related to Runtime Infrastructure - Feature #6364: implement an option to use...	**New**
Related to Base Language - Feature #6343: class-based appserver exports	**New**

History

#1 - 05/17/2022 10:35 AM - Greg Shah

- Related to Feature #4065: server-side processing of client platform dependencies added

#2 - 05/17/2022 10:35 AM - Greg Shah

- Related to Feature #6364: implement an option to use Tomcat instead of Jetty as the embedded webserver/servlet container in the FWD server added

#4 - 05/17/2022 10:37 AM - Greg Shah

- Related to Feature #6343: class-based appserver exports added

#5 - 05/17/2022 10:47 AM - Greg Shah

A customer with very heavy appserver usage currently runs their Open Java Client code in a separate Tomcat JVM. This means that there is quite a bit of extra serialization/deserialization overhead per appserver call (2 serializations + 2 deserializations for each round trip). They want to eliminate this overhead and combine this with additional changes:

- Use #6364 to optionally allow this Open Java Client code to run in process in the FWD server.
- Use #4065 to eliminate the need for a client JVM per agent and to speed up all the client platform usage (e.g. memptrs, library calls, sockets, file access).
- Use #6343 to allow objects to be exposed via appserver. The would change their Open Java Client code to use objects and bypass the need for all their generated procedural stubs. This will eliminate a layer on each side of the appserver call.

The idea of this task is to implement the ability to have a local proxy to the #6343 objects that requires no serialization/deserialization. This would allow some amount of the appserver processing to be completely bypassed, greatly reducing overhead. The same OO approach would be possible in both remote and local proxies, but the remote version would need the normal serialization/deserialization while the local proxy would avoid it.

They would write the middle tier code to get a local proxy when they run in-JVM on the FWD server and would get a remote proxy (normal #6343 case) when they run in a separate JVM.

#6 - 06/15/2022 05:37 PM - Greg Shah

Will the local proxy submit a "job" to the agent pool for a given appserver OR will we be able to call the converted code "in place"? The "in place" idea is the same as #4406 but for appserver, not REST. This seems the best outcome but it has the same potential issues.

If we go for the "in place" option, then we also must consider how we deal with the multiple appservers issue (it is common for more than one named appserver to be defined in OE, with different propaths/configuration, each appserver potentially servicing different "applications").

A related problem is how to eliminate the overhead of cross-appserver calls. This is a call to one appserver from another, within the same FWD server.