**IMS implementation details v1.0**

*Overview*

The basic idea behind IMS is to provide an ability for objects in the application to communicate with each other using messages, instead of direct method calls.

Every object has an id by which it can identified in IMS. Then knowing that id we can send messages to the object using intermediate objects – routers and channels.

The pros of this are

* Unification – accessing the fields, setting the data and etc. is all the same in the scope of the application runtime
* Higher flexibility as we are not bound to object expectations regarding the result
* Ability to divide objects into groups with different addresses, combining object groups etc.

The cons are

* Performance drop – as we are not invoking anything directly we get huge overhead due to a lot of method invocation during redirect from one object point to another.

IMS is similar to JMS but unlike the latter, it does not provide session and does not have specific requirements as I made out of interest.

*Architecture*

The architecture diagram presented below.

Every object is associated with a specific address – global resource locator or GRL. GRL has format

*a:b:c*

For example

*core:component:componentStorage*

Messages that object send are called GRL messages and they has GRLMessage type. Message structure looks like this

|  |  |
| --- | --- |
| METHOD | message type |
| HEADERS | map of the headers supported by IMS |
| MULTIPART | any object content – json, binary etc. |

Methodis an object of GRLMethod type. Currently supported three methods – POST, PUT, GET.

1. POST – set object value or update
2. PUT – updated object value (I think it can be removed)
3. GET – returns object value

Headers is a map with multiple key values, where keys are supported headers. Headers might be required and might be not required. During the sending of message this is being checked.

|  |  |
| --- | --- |
| **Header** | **Required** |
| destination | True |
| sender | True |
| receiver | True |
| contentType | False |
| sendTime | False |
| receiveTime | False |
| brokerTime | false |

Multipart is an implementation of IGRLMultipart interface, which however can be used to store any type.

|  |  |  |
| --- | --- | --- |
| Interface | IGRLMultipart |  |
| Methods | getContent() | Returns wrapped object content |