# Avatar Development Environment

Purpose: I2WD is working with University of Delaware on a new I2WD program call Avatar, which is an intelligent aid for the warfighter. An Avatar briefing has previously been approved for public release and shared University of Delaware personnel.

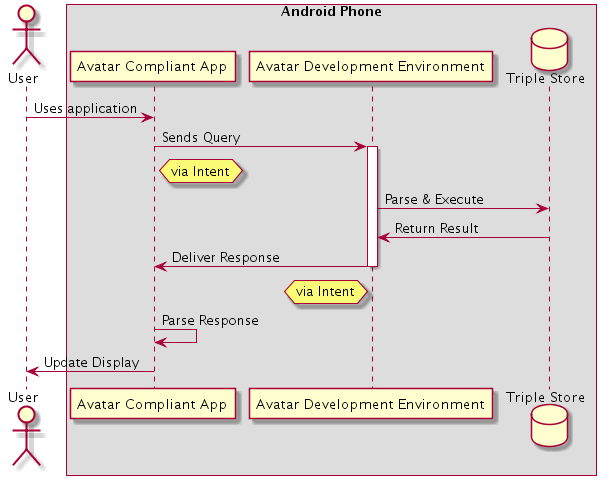
Introduction: This document includes basic instructions (i.e. documentation) for working with the beta software known as the Avatar Development Environment (ADE)(ADE File name is ‘Avatar Software.7z’). This software and documentation will be provided to approximately 4-8 students at the University of Delaware as part of a senior capstone project I2WD plans to sponsor. I2WD personnel will also help these students work with this software.

The Avatar Development Environment (ADE) is an Android application that works in conjunction with a custom version of Apache Jena (<https://jena.apache.org/>). ADE is built to work on the Android Software Development Kit (SDK) version 19.

The ADE also has an embedded Android Intent Service (<http://developer.android.com/reference/android/app/IntentService.html>), which can be called, via a service call, by other applications on the phone to expose Jena's Resource Description Framework (RDF) capabilities.

This embedded Intent Service is an asynchronous, queued service that responds to Intents in the order they are received, and will broadcast a Response Intent once the action has been processed.

Below is a flow diagram of the general use case.



Distribution A: Approved for public release; distribution is unlimited

## System Configuration

When started, the ADE will create its own RDF store directories. The only thing that you will need to add is the Ontologies you would like to load. These should be placed in the storage location as follows: [/storage/emulated/0]/Android/data/org.cubrc.avatar/Ont

The “/storage/emulated/0” part of the file path represents the Android phones ExternalStorageDirectory, this may change from phone to phone.

Below are the actions that the ADE supports:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Action URI** | **Description** | **Payload Type** | **Return Type** | **Notes** |
| org.cubrc.avatar.SCAN | Lists all of the triples in the model | N/A | ArrayList<ArrayList<String>> | Payload represents a List of Rows<List of Columns<Value>> |
| org.cubrc.avatar.TRIPLES | Inserts the given triples into the RDF store | ArrayList<Bundle>  See below | String | "Triples Inserted" returned if successful |
| org.cubrc.avatar.SELECT | Executes a SPARQL Select Query | String - Query | ArrayList<ArrayList<String>> | Payload represents a List of Rows<List of Columns<Value>>  First row is the Header row (contains the SPARQL Bind variables) |
| org.cubrc.avatar.CONSTRUCT | Executes a SPARQL Construct Query | String - Query | ArrayList<ArrayList<String>> | Payload represents a List of Rows<List of Columns<Value>> |
| org.cubrc.avatar.ASK | Executes a SPARQL Ask Query | String - Query | Boolean |  |
| org.cubrc.avatar.UPDATE | Executes a SPARQL Update Query | String - Query | String | "Update Executed" returned if successful |
| org.cubrc.avatar.PURGE | Removes all statements in the model  This does not remove ontologies | N/A | String | "Datastore Purged" returned if successful |

Note: Due to the potentially large nature of RDF Models with loaded ontologies, Actions that return ArrayList payloads will be limited to only returning 100 rows.

To send one of the above Actions, simply create an Intent with the desired Action Uniform Resource Identifier (URI), and use the same URI as an extra to attach your payload.

String query = "SELECT ?s ?p ?o WHERE { ?s ?p ?o . }";

Intent selectIntent = new Intent("org.cubrc.avatar.SELECT");

selectIntent.putExtra("org.cubrc.avatar.SELECT", query);

selectIntent.putExtra("org.cubrc.avatar.REQUEST\_ID", "org.cubrc.avatar.samplerdfactivity.QueryFragment.Select"); //Optional

startService(selectIntent);

Note in the above sample, the fourth line added an additional ‘putExtra.’ This is because by nature Intents are broadcast throughout the Android platform, and all ADE compliant applications will be listening for their own Response Intent.

This REQUEST\_ID field will be copied to the Response Intent so applications can filter out service responses they do not handle or initiate.

### Action: Triples - Payload Structure

This section will provide the developer with an understanding of the required payload structure of the ‘Triples’ action.

It is recommended that you use SPARQL queries when creating Triples; however, the ADE supports the ability to submit an ArrayList<Bundle> (<http://developer.android.com/reference/android/os/Bundle.html>) to generate Triples.

Each Bundle in the ArrayList represents a Triple statement to insert. All values in the Bundle Map are Strings. All values are assumed to be URIs unless you specify otherwise.

|  |  |  |
| --- | --- | --- |
| **Key** | **Value** | **Notes** |
| "SUBJECT" | (subject URI) |  |
| "PREDICATE" | (predicate URI) |  |
| "OBJECT" | (object URI or Literal) |  |
| "SUBJECT\_TYPE" | "SUBJECT\_TYPE\_URI"  "SUBJECT\_TYPE\_BLANK" | Optional |
| "OBJECT\_TYPE" | "OBJECT\_TYPE\_URI"  "OBJECT\_TYPE\_BLANK"  "OBJECT\_TYPE\_LITERAL" | Optional |

## Handling Action Response

After the ADE compliant application submits an Action to the ADE, the request will be executed in an asynchronous manner. Once this job has completed, the ADE will produce an Intent in response.

The service will respond to the Action by broadcasting an Intent with the Action URI "org.cubrc.avatar.BROADCAST". The ADE compliant applications will need to set up an Intent Filter and Broadcast Receiver to intercept this response in order to get results.

Below is the mapping of the Broadcast Response Intent's Extras.

|  |  |
| --- | --- |
| **Key** | **Value** |
| "org.cubrc.avatar.TYPE" | "org.cubrc.avatar.SELECT" (will match the request action) |
| "org.cubrc.avatar.STATUS" | "DONE" or "FAIL" |
| "org.cubrc.avatar.REQUEST\_ID" | "org.cubrc.avatar.samplerdfactivity.QueryFragment.Select" (will match request intent request id, if provided) |
| "org.cubrc.avatar.RESPONSE" | Type appropriate request action (from above table)  OR  If STATUS == "FAIL", will be a String explaining what went wrong, regardless of Action |