JTLS-2016-12705 Robust Filtering For AAR Collection

Rick Kalinyak

1.0 Summary of Model Change Request

The JTLS After Action Review Client (AARC) is responsible for collecting AAR objects from the JODA and inserting the information into database tables. This information includes objects, object events, and interactions. For interactions, such as object detections and attrition events, the amount of information can become extremely large, slowing down the saving of data during checkpoints.

Currently, in an attempt to limit the amount of data saved, a filtering capability exists. This filtering capability can turn off the saving of information to specific tables, or can be used to turn off the saving of specific object events. No capability exists for robust filtering, which might include only saving interaction information if one of the objects involved belongs to a specific force side. For example, we only care about detections of ALLIANCE and OPFOR units, not target detections or detections of units on other sides. This ECP outlines a filtering capability to allow user-definable robust filtering of AAR objects.

2.0 Design Summary

The current filtering mechanism could be expanded to provide desired capabilities, but ultimately a capability that is definable based on information from the JODA objects provides the most robust possibilities for filtering. In this way additions or changes to the AAR data will not necessitate new code to support filtering.

This type of filtering already exists in JTLS for object selection within orders. The order filters provide for the specification of an object type, and comparison of any attribute of the object, including compound object references. Figure 1 shows the filtering for the Highres Unit field in a Manage HRU Tasks order, which requires that the unit be an active HRU owned by JTLS and for which the player has command authority over the HRU's parent unit.

```
<fill using="name">
   <with types="hru">
      <not-equal a="UNIT.cmd_parent_id.command_authority" b="NONE"/>
      <and/>
      <is-true a="UNIT.is_active"/>
      <and/>
      <equal a="UNIT.side" b="THIS_SIDE"/>
      <and/>
      <equal a="UNIT.external_model" b="JTLS_ONLY,JTLS"/>
   </with>
</fill>
```

FIGURE 1. Highres Unit Field Filter From Manage HRU Tasks Order

Figure 1 provides an example of an order filter:

- Utilizing a compound JEDI comparison ("UNIT.cmd_parent_id.command_authority"),
- A boolean attribute check (<is-true a="UNIT.is_active"/>),
- An enumerated attribute check against an enumeration (<equal a="UNIT.side" b="THIS_SIDE"/>), and
- An enumerated attribute check against an enumeration list (<equal a="UNIT.external_model" b="JTLS_ONLY,JTLS"/>)

Most, but not all, of the filter configuration within Figure 1 will be directly used by the AAR. The parts that are not relevant are the <fill using=””> tag, which specifies which attribute of valid objects should be entered on the order, and the UNIT keyword which is merely a marker for the order field.

An example of the proposed AAR filter schema appears below in Figure 2.

```
<table name=”aar_unit">
  <with types="aar_unit">
     <not-equal a="cmd_parent_id.name" b="HQCENTCOM"/>
     <and/>
     <not-equal a=”name” b=”HQCENTCOM”/>
     <and/>
     <equal a="side" b="SIDE_1,SIDE_2,SIDE_5"/>
  </with>
</table>
```

FIGURE 2. Example Portion AAR Filter

The example AAR filter in Figure 2 is for the AAR_UNIT table, which is also the name of the object used to fill the table in the JDS Protocol. The filter specifies that the object inserted must be of type aar_unit, that it can not be HQCENTCOM or a direct subordinate of HQCENTCOM, and that it must be on sides 1, 2, or 5. The name, side, and cmd_parent_id are all attributes of the AAR_UNIT object within the JDS Protocol. It might seem redundant to specify that the aar_unit table can be filled with objects of type aar_unit, but by listing it twice we provide the capability to specify an empty types attribute in which case no objects would pass the filter and be inserted into the table.

An interface for the Technical Controller will be provided for the generation and modification of the filter file. This interface will be incorporated into the Interface Control Program (ICP), allowing the Technical Controller to turn on/off individual tables, or setup the robust comparison filters.

3.0 Detailed Design

3.1 Format of the Filter File

Every AAR JODA Object, which corresponds to an AAR table, will have an entry in the filter file, similar to the AAR_UNIT table entry in Figure 2.

If the collection of data for the table is turned off, the types attribute of the <with> tag will be empty. If the table is turned on the types attribute will have the same value as the table name. If all such data is to be saved there will be no comparison tags. Turning on the table and specifying comparison tags will restrict the JODA objects that will be inserted into the AAR tables.

Table 1 displays the recognized comparison tags and the meaning of the two attributes.

Table 1. Filter Comparison Tags

COMPARISON TAG	ATTRIBUTE A	ATTRIBUTE B
equal	An attribute or compound attribute of the JODA object.	An attribute or compound attribute list of the JODA object, or an explicit value list.
	<equal a="side" b="SIDE_1,SIDE_2,SIDE_5"/>	
not-equal	An attribute or compound attribute of the JODA object.	An attribute or compound attribute list of the JODA object, or an explicit value list.
	<not-equal a="cmd_parent_id.name" b="HQCOM"/>	
is-true	A boolean attribute or compound attribute of the JODA object.	N/A
	<is-true a="object_id.alert_flag/>	

Table 1. Filter Comparison Tags

COMPARISON TAG	ATTRIBUTE A	ATTRIBUTE B
is-false	A boolean attribute or compound attribute of the JODA object.	N/A
	<is-false a="alert_flag"/>	
greater-than	An attribute or compound attribute of the JODA object.	An attribute or compound attribute of the JODA object, or an explicit value.
	<greater-than a="latitude" b="50.2"/>	
greater-than-or-equal	An attribute or compound attribute of the JODA object.	An attribute or compound attribute of the JODA object, or an explicit value.
	<greater-than-or-equal a="longitude" b="50.0"/>	
less-than	An attribute or compound attribute of the JODA object.	An attribute or compound attribute of the JODA object, or an explicit value.
	<less-than a="latitude" b="longitude"/>	
less-than-or-equal	An attribute or compound attribute of the JODA object.	An attribute or compound attribute of the JODA object, or an explicit value.
	<less-than-or-equal a="longitude" b="9.0"/>	

As specified in Table 1,

- Attribute “a” of a comparison tag will always be an attribute or compound attribute of the JODA object.
- Attribute “b”, if it is used, can be an attribute or compound attribute of the JODA object, or can be an explicit value.

The AAR Filter Interface (AFI) will be responsible for ensuring the modes of attributes “a”, and “b” match. However, if the filter file has been hand-edited and the modes do not match then the comparison will always fail (meaning a string is neither less than, equal, or greater than the numeral 0.0). String comparisons will be based on their English language alphabetic precedence, numeric comparisons on their numeric value, enumerated values on their equivalent integer value, and boolean comparisons on their 1/0 value.

The is-true and is-false comparisons are designed for booleans, but in the event another attribute type is entered any non-empty string is true, any non-zero enumerated or numeric value is true.

In addition to the condition checks specified in Table 1 the syntax also supports the conjunctions specified in Table 2

Table 2. Filter Conjunction Tags

CONJUNCTION TAG	MEANING
and	A logical and conjunction. <not-equal a="cmd_parent_id.name" b="HQCENTCOM"/> <and/> <not-equal a="name" b="HQCENTCOM"/>
or	A logical or conjunction. <not-equal a="name" b="HQCENTCOM"/> <or/> <equal a="side" b="SIDE_1,SIDE_2,SIDE_5"/>
all	The same as a set of parenthesis in a logical operation, where the conditions within the "all" tag should be evaluated first to provide a single result. <all> <not-equal a="cmd_parent_id.name" b="HQCENTCOM"/> <and/> <not-equal a="name" b="HQCENTCOM"/> </all> <or/> <equal a="side" b="SIDE_1,SIDE_2,SIDE_5"/>

3.2 AAR Filter Configuration Interface

The AAR Filter Interface (AFI) is responsible for creating an initial filter file, reading an existing file, allowing changes to the filters, and writing the filter file.

The AFI is the third program in which special expanded configuration files are required. The other two programs are:

- The Message Deliver Program (MDP) configuration definition.
- The JTLS Satellite Service (JSAT) configuration definition.

The Design Team feels that in the future the number of such programs could continue to expand. For this reason it makes more sense to have all of the configuration capabilities under a single configuration program, which is the Interface Control Program (ICP). For this reason, ICP will be modified to access the new AFI from the ICP interface.

The ICP changes will be made to easily allow additional configuration interfaces for the MDP, JSAT and other future program requirements. Expanding the ICP for the MDP and JSAT is outside

the scope of this ECP, but the current plan calls for using JS/J7 funding to complete the ICP configuration responsibilities. Adding the MDP and JSAT configurations to the ICP is not needed for the delivery and acceptance of this ECP. This discussion is simply provided so the reader can understand the final goal for the ICP.

The AFI automatically determines the AAR objects and their attributes by reading the JDS Protocol, which has the AAR objects within their own section. A list of tables will be shown and the AFI will allow tables to be filtered on or off via a toggle button. Highlighting a table shows the existing filters applied to the table and allows additional filters and conjunctions to be added.

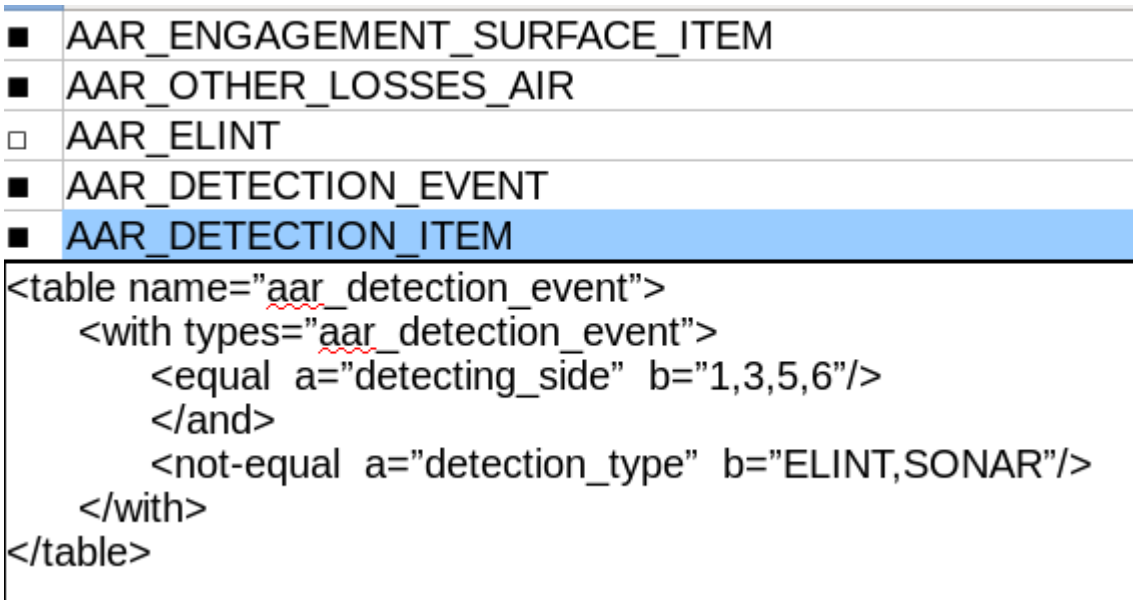

FIGURE 3. AAR Filter Interface

Figure 3 shows a top level interface for the AFI, with a list of tables in the top section, the AAR_DETECTION_EVENT table selected, and the filters for that table displayed in the lower half. If the AAR_DETECTION_EVENT table was toggled off in the upper section then the with/types attribute in the lower section would be empty.

The creation of the filter definition in the lower section will not be hand-edited but will be driven by a selection process. The user starts by either selecting an attribute of the JODA object, or selecting the <all> tag. If the selected attribute is a JEDI, all other attributes within the AAR objects, not just attributes of the current object, are displayed and can be added. This includes the unlisted attribute object_class which will decode the object_class from the JEDI. Whenever a non-JEDI attribute is chosen the user may select the comparison operator, and then based on the mode of the attribute may select the item to be compared against.

Line-by-line contents of the table filter will be selectable, and may be deleted. And/or conjunctions can not be deleted, but whenever a comparison that follows a conjunction is

deleted the conjunction will also be deleted. If an all conjunction is deleted, only the all tags will be deleted, the comparisons within the all tag will remain and must be deleted individually.

There is a danger that the Technical Controller creates an interface file that will be inconsistent, for example not saving any air missions but saving air mission event data. This will result in a foreign key violation when attempting to insert the air mission events. Since the JDS Protocol does not contain information about these relationships it is not the responsibility of the AFI to prevent them. Instead the AARC will be expanded to check for inconsistencies after reading the filter file and generating error messages for such inconsistencies. This will not prevent the filters from being used, but will inform the Technical Controller of the situation so that they may go back to the AFI and modify the filters.

3.3 AAR Reports

By limiting the data that is inserted into the AAR tables this will also limit data that can be reported via the TRIPP/AAR Reports. While this statement might seem obvious there is a concern about the TRIPP operators, who have not set up the filters, requesting a TRIPP/AAR report and not seeing all the results reported by the model. This will likely lead to confusion and questions about the validity of data within the AAR tables. In order to mitigate this situation the AARC will save the filters as part of the database whenever a new filter file is read. This will permit the TRIPP/AAR reports to extract the proper time-stamped filter settings and deliver the relevant portions as part of the report.

3.4 CEP Changes

The CEP will require two modifications for the AAR data it sends to the JODA.

1. AAR object references are currently being sent using just the object's receiver number and not encoding the object class for a full JEDI. This will be modified so every time an object reference is sent by the CEP the JODA packet will encode an object class.

2. A new AAR object delete structure will be created for detection and engagement events.

   Currently both the AAR_DETECTION and AAR_ENGAGEMENT_SURFACE tables record the detection/damage event but not the results of the event. The results are recorded in children tables, AAR_DETECTION_ITEM and AAR_SURFACE_SURFACE_ITEM respectively. This parent-child relationship is not outlined in the JDS Protocol, from which the majority of the AARC code is generated. The AARC will keep track of objects, even if they have not been inserted into the database because they might be part of a compound JEDI filter check. However, we do not want the AARC to maintain millions of detection and engagement events. Therefore, the new destroy packet will be a signal from the CEP that no further references to the destroyed object will be sent, and the AARC can terminate tracking it.

3.5 Performance Considerations

Currently the AARC simply receives JODA objects, and if the corresponding table or event is not filtered off, the data is inserted directly into the AAR database tables. This level of filtering is quick and easily accomplished. With the more robust filtering specified in this design a filter could become very complex and time consuming to evaluate slowing the insertion of records into the database tables. This is not an expected concern when running at normal exercise speeds (up to 4:1), but could be an issue when running at faster than normal speeds. The data will not be lost, but the AARC could fall behind the CEP model, catching up when the game is slowed or during periods of lesser activity.

4.0 Data Changes

No data changes are required for this ECP.

5.0 Order Changes

No order changes are required for this ECP.

6.0 JODA Changes

The JODA AAR objects do not currently use proper JEDI attributes, but instead use an unsigned 32-bit integer to hold the object number with no object-class encoded into it. These numbers will be changed to proper JEDIs which encode both the object class and object number. The biggest issue for this is parameters within events sometimes contain object numbers, and these will need to be identified and modified to contain a proper JEDI.

7.0 Test Plan

TBD. Depends on which aspects of the design are changed/approved.

7.1 Test 1 Title

Purpose: [*Describe the specific feature, function, or behavior to be tested or measured.*]

Step 1: Text

Step 2: Text

Expected Results: [*Describe the specific model behavior to be observed.*]

7.2 Test 2 Title

Purpose: [*Describe the specific feature, function, or behavior to be tested or measured.*]

Step 1: Text

Step 2: Text

Expected Results: [*Describe the specific model behavior to be observed.*]