**ASRS Tech Id for RAI**

The existing RAI interface currently assumes that the tech Id will be “<ASRS>”. Now that we have introduced XML/sockets using the integrator for RAIng, the tech Id is set by the RTCIS session and may not be “<ASRS>” (i.e. configurable). In addition, we may have two (or more) systems/sessions sending CSI/RAI transactions to RTCIS, and they each system/session may have a different tech Id.

There are several places in the RTCIS code base which identify objects (shipments, truck load ULs, etc) that are assigned to a normal tech/user or assigned to the ASRS tech/user. The code currently just checks for the “<ASRS>” token (defined by the AP\_ASRS\_TECHID macro) to determine if the object was last assigned to the ASRS tech/user or a normal tech. This causes bugs in the RAIng interface when the tech/user set by integrator session is not “<ASRS>”. This document contains suggestions to fix this bug.

*Note: Each of the approaches described below require adding an asrs\_user\_number field to dm\_c.*

**Approach 1: Hard-code the tech Id to “<ASRS>”**

The simplest solution would be to hard code the tech/user to “ASRS” in each ASRS integrator transaction (used by RAIng) and each “act” program (used by RAI 1.0). *Note: RAI 1.0 assumed that the DM\_USER environment variable was set to “<ASRS>” before the “act” program was invoked. The script in production that invokes all “act” programs does ensure this, but this causes issues for testing RAI 1.0, when the “act” program is invoked from the command line.*

**Pros:**

* Zero impact/risk on non-RAI application.
* Zero impact/risk for RAI 1.0.
* Minimal development effort (roughly 2 hours)

**Cons:**

* RTCIS will not track individual systems/techs when multiple systems send transaction.
  + All objects will be owned by the generic “<ASRS”> user.
  + All history will be logged by the generic “<ASRS>” user.
* Creates another limitation for supporting multiple ASRS systems in the future.
* Most limiting approach for RAIng.

**Approach 2: Hard-code the tech Id to “<ASRS>”, but record the session tech Id in history**

A slightly more flexible solution would be to save the system/sessions tech/user and temporarily set the tech/user to “<ASRS>” in each ASRS integrator transaction (used by RAIng). Each “act” program (used by RAI 1.0) would still hard-code the tech/user to “<ASRS>”. RTCIS would use the saved tech/user when logging history, such as system transactions or message history.

**Pros:**

* Zero impact/risk on non-RAI application.
* Minimal impact/risk for RAI 1.0.
* All history will be logged by the real system/session tech/user.
* Minimal development effort (roughly 8 hours)

**Cons:**

* RTCIS will not track individual systems/techs when multiple systems send transaction.
  + All objects will be owned by the generic “<ASRS”> user.
  + Potential confusion when comparing the object’s tech/user to the history tech/user.
* Creates another limitation for supporting multiple ASRS systems in the future.

**Approach 3: Dynamically determine which tech Ids are automation tech Ids**

All automation using CSI transactions (including RAI) must specify an RTCIS session Id. Therefore, we can safely assume all statically configured RTCIS sessions (as defined in the RTCIS\_SESSION table) represent automation users. We can modify the RTCIS code (that currently checks for the “<ASRS>” user) to outer join to the RTCIS\_SESSION table. If the row is found, RTCIS will treat the user as an automated user. If the row is not found, RTCIS will treat the user as a normal tech. The logic in each routine that processes automated user v. normal techs would be unchanged (beyond how to determine the type of user).

Each “act” program (used by RAI 1.0) would still hard-code the tech/user to “<ASRS>”. The “act” program would then check if there was an entry in the RTCIS\_SESSION table for the “<ASRS>” user and create one, if needed. Creating the RTCIS\_SESSION entry for RAI 1.0 would be required, so that the modified RTCIS code finds an RTCIS\_SESSION, so that it may identify the program as an automated user.

*Note: I also considered adding a flag to the USERS table, to indicate if the user was an automated user. Dan pointed out that there would be complications requiring automated users to be in the USERS table, because these could be periodically purged by security audits and that we do not want to create an OS user for automated users. So, I suggested that we could outer join RTCIS\_SESSION instead.*

**Pros:**

* Each RTCIS object will be tracked by the proper system/session tech/user.
* All history will be logged by the real system/session tech/user.
* Flexibility for building toward supporting multiple ASRS systems in the future.
* No table changes required.
* Reasonable development effort (roughly 20 hours)

**Cons:**

* Potential impact/risk on non-RAI application, but risk is low.
* Potential impact/risk for RAI 1.0, but risk is low.
* Assumes RTCIS\_SESSION user will not be changed while RTCIS is running with objects assigned.
* Extra table join to determine user type.

**Approach 4: Statically track objects that are assigned to automation tasks**

The most comprehensive solution would be to add a column on to every affected table, indicating that the object was assigned to an automation user. The following tables would need this new column:

* Matreq for the techid column
* Replen for the techid column
* Shphdr for the ultch2 column
* Trukld for the techid column
* Wdsubr for the cantch column

We can modify the RTCIS code (that currently checks for the “<ASRS>” user) to check for the new column. The logic in each routine that processes automated user v. normal techs would be unchanged (beyond how to determine the type of user).

We would also have to modify all RTCIS code that inserts a new row or updates the tech Id on an existing row. *Note: That there are many more routines that insert/update than there are routines that check the user type (automated v. normal user) in the RTCIS code base.*

Each “act” program (used by RAI 1.0) would still hard-code the tech/user to “<ASRS>”.

**Pros:**

* Each RTCIS object will be tracked by the proper system/session tech/user.
* All history will be logged by the real system/session tech/user.
* Flexibility for building toward supporting multiple ASRS systems in the future.

**Cons:**

* Potential impact on non-RAI applications.
* Potential impact on RAI 1.0.
* Five table changes required.
* Maximum development effort (current guestimate is about two weeks).