tesserNet SYSTEMS INC.

RTAP Database Lifecycle Management

Abstract: This paper will describe the typical RTAP database lifecycle and how the tesserNet Systems Inc. tools can make this process easier to manage and less error prone. This paper assumes the reader is familiar with the RTAP product and its database.

Contents

RTAP Database Lifecycle

One of RTAP's major strengths is its database. The RTAP database allows integrators and users to customize its structure to fit their exacting requirements. Because of this flexibility, there can be some effort required to maintain and create databases.

The typical RTAP database lifecycle consists of the following steps:

1. *Design RTAP point classes.*
2. *Create database using point classes.*
3. *When design changes occur, make changes to the existing point classes or create new point classes and then correct the existing points in the database.*

This paper will address each of these lifecycle steps to see how they can be better managed and accomplished using tesserNet Systems Inc. tools.

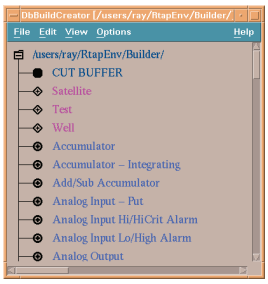
RTAP Point Class Design

The typical RTAP database point class design requires the user to determine what point classes are required and the functionality that each of these point classes will provide. The functionality then determines the attributes that will be created as part of each point class. Creating these point types is usually done by either creating a template point in the database, or by creating an off-line point configuration file.

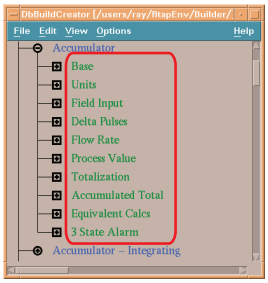
The tesserNet **dbBuildCreator** tool provides a graphical interface that is used to build the various point classes. The configuration files created by this tool are ASCII files. These files can be moved or copied between various UNIX machines without any conversions. This allows cross platform re-use.

The dbBuildCreator tool provides an additional level of abstraction within the RTAP database structure with the use of "blocks". Each "block" is a group of one or more RTAP attributes. This grouping of attributes allows the designer, and ultimately the user, to concentrate on a particular set of attributes. For example, a "Scale" block may consist of the following set of attributes used to scale a raw value from the RTU:

- *raw min value (rtDouble)*
- *raw max value (rtDouble)*



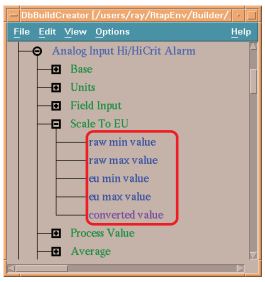
dbBuildCreator tool.



Blocks provide an additional level of indirection.

- *eu min value (rtDouble)*
- *eu max value (rtDouble)*
- *converted value (rtDouble)*

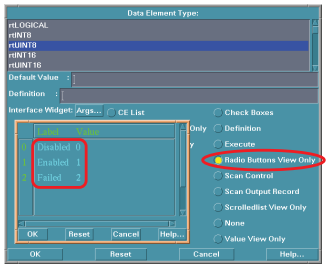
By grouping these attributes together, database maintenance personnel are able to easily view these attributes together when needed.



Grouping of attributes into blocks.

The use of "blocks" also allows the database designer to re-use these attributes in multiple point classes to maintain a consistent set of attribute names associated with this functionality. This re-use allows for faster point class development and easier maintenance, since changes in the block will be reflected in all point classes that use that block. This functionality is commonly called "multiple inheritance" in object oriented terminology.

The dbBuildCreator tool also provides the ability to specify how the various attributes will be presented to the user, and additional on-line help functionality. The ability to specify the attribute presentation, allows the database designer to provide a more intuitive interface for the user. For example, in the communication port point in the scan system, an attribute called "status" exists that represents the current status of the port. This attribute can have one of three values:

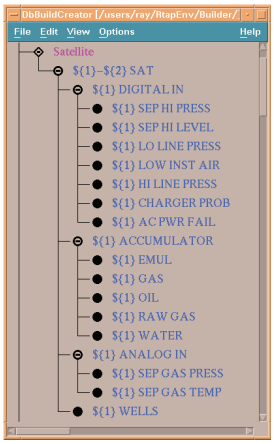


Control of attribute display properties.

- *0 = disabled*
- *1 = enabled*
- *2 = failed*

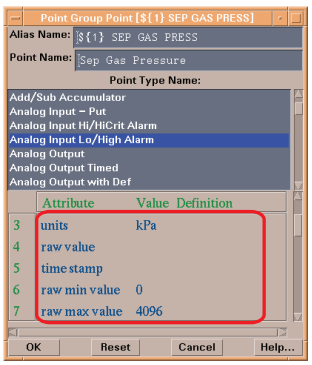
This attribute should not be changed by the user and should be treated as a read-only attribute since it is maintained by the scan system. Using the dbBuildCreator tool, the database designer can choose to present this attribute as a set of read-only radio buttons that describes the actual value. In this case, the user doesn't have to remember that zero means that the port is disabled, one means that the port is enabled, and two means that the port is failed.

Once point classes have been defined, it is sometimes beneficial to group these point types into a well defined database hierarchy that represents a physical device. For example, an oil well for company "xyz" will always consist of a fixed set of alarm points, a fixed set of analog values, a fixed set of digital control points, and a fixed set of accumulator points. The dbBuildCreator tool allows for the creation of "point groups" that model the physical database hierarchy using the existing set of point classes. String substitution provides for unique alias names as well as custom values within the points themselves. These "point groups" provide consistent naming of points for like branches.



Point groups allows for the creation of database branches of an arbitrary number of points and depth.

The "point groups" allow the database designer to customize instances of the point classes. For example, there maybe a generic "analog input" point class that can be used for any analog input value. When creating a "point group" the database designer can use this generic "analog input" point class for a specific analog and customize it accordingly. For example, the generic "analog input" point class can be used for an incoming pressure value. It can then be customized so that its unit is set to "kPa", its raw minimum scaling factor is set to zero, its raw maximum scaling factor is set to 4095, its engineering units minimum is zero, and its engineering units maximum is 1000. This results in less error when the "point group" is created in the database.



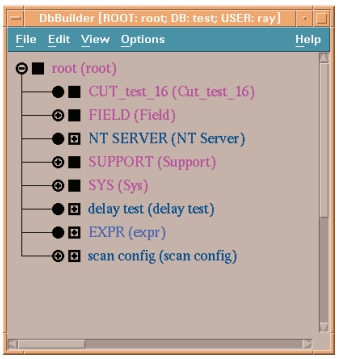
Point groups also allow for customization of point class data.

RTAP Database Creation

The typical database creation is performed either manually by copying a database template point, or by modifying a point configuration file and loading it into the database.

The tesserNet **dbBuilder** tool provides a single interface for the creation and maintenance of an RTAP database. Using this tool, the user is able to create individual database points from the point classes or "point groups" defined by the dbBuildCreator tool. This tool ensures that each point created from a defined point class is consistent (no extra attributes, no missing attributes, and consistent attribute names).

The dbBuilder tool also provides the user with the view into the database points, grouped by "blocks". This way the user doesn't have to search a list of up to 255 for the attributes that they are interested in.

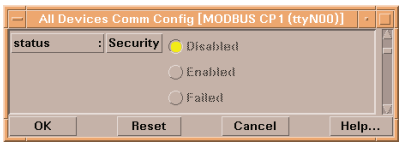


Database builder tool.

When making changes to the RTAP database using the dbBuilder tool, it is possible to have dbBuilder connect to multiple RTAP database instances so that the changes are made to all databases at the same time. This is useful when multiple database instances are used for redundancy.

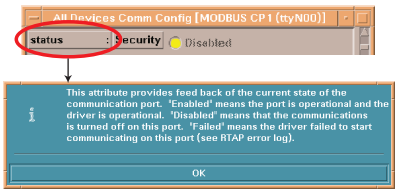
Some of the features that dbBuilder provides are:

- *A single graphical interface into the RTAP database (benefit: only a single program to run to configure the database, adheres to the Motif/ X11 graphical interface so users do not have a steep learning curve).*
- *Standard "cut", "copy", and "paste" functionality for database points (benefit: users are already familiar with these operations from other applications; no additional training required).*
- *Widget based display of attribute values (benefit: users do not have to remember what attribute values actually represent (e.g. 0 = disabled), or what the valid values are for an attribute (e.g. a list of possible units can be displayed ("kPa", "feet", etc.) in a selection list, users then don't have to remember the correct spelling for each one and the database will have consistent values)).*
- *Insertion/deletion of records/fields in tables (benefit: on-line modifications to tables).*
- *Handles all of the underlying API calls required to perform an action (benefit: users don't have to remember what **else** is required to perform a task; for example, disabling a communication port requires an API call to be made to the scan system).*
- *Easy creation of a database branch hierarchy (point groups) (benefit: after entering values into a single dialog, a whole database branch, possibly containing hundreds of points, will be created; a time saving for the user).*
- *Detailed database editing using rtsh scripting (benefit: easily apply the same database changes to a set of database points; a time saving for the user).*
- *Editing of multiple redundant databases at the same time (benefit: keeps multiple databases in sync).*
- *On-line help for attributes (benefit: users do not have to remember what a particular attribute is used for).*



Displays attributes using display properties defined in the dbBuildCreator tool.

The dbBuildCreator tool allows the database designer to specify some help text and an interface widget for each attribute. The dbBuilder tool serves this up to the user providing for a more intuitive interface and if required, additional help for the attribute.

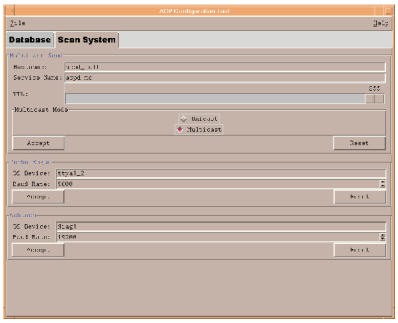


Selecting the attribute push button displays on-line help for that attribute.

The tesserNet **rtsh** and **rtperl** tools provide scripting capabilities for database creation and modifications. Both of these tools provide hooks into the majority of the RTAP API calls and are not limited to just database functions. For example, to disable a communication port in the scan system, the user should not be setting the "control" attribute in the communication port point to zero. Instead, they must make the appropriate scan system call to the scan manager to disable the communication port. This can be accomplished using these scripting tools as follows:

- *rtsh* – `controlSs action=DISABLE_CP name="<alias>MY COMM PORT"`
- *rtperl* – `rtapSs->disableCp ("<alias>MY COMM PORT");`

In addition to the RTAP API calls, these scripting tools also provide calls to create a point class or "point group" as previously defined using the dbBuildCreator tool. This allows the user to create scripts to create a whole database, or just a branch.



Using rtperl and the tk perl module allows for sophisticated graphical scripts.

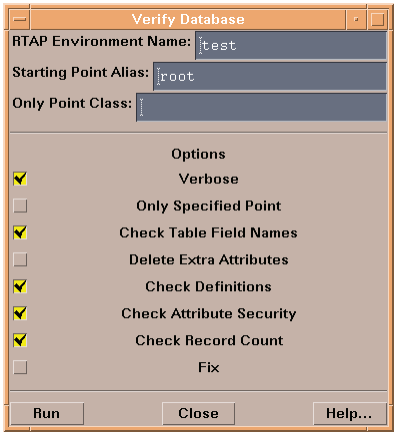
The rtsh tool provides a simple shell scripting interface. The shell scripting language is simple to learn, but provides sophisticated capabilities such as flow control (if then else), looping (while for loops), and custom functions.

The rtperl tool provides a more sophisticated language that is similar to the C programming language. It also provides a multitude of other perl modules that can be freely downloaded. One of these modules is the "tk" graphical interface. Using rtperl and the "tk" module (included as part of the rtperl installation), users are able to create sophisticated graphical scripts that are able to interact with the user.

RTAP Database Modifications

As much as database designers would like, database point classes do not remain static for very long. Scope changes, and functionality changes result in modifications to existing point classes.

Using the tesserNet tools, there are several methods to implement database changes. Initially the change is made to the point class using the dbBuildCreator tool. Depending on the nature of the change, from within dbBuildCreator, the user is able to "verify" the change to a RTAP database instance. This verification allows the user to ensure that the points in the database match the structure that has been defined by dbBuildCreator. Due to the fact that there could be dependencies within the database, the "verify" may not be able to fully synchronize the database. For example, if an attribute is used in another attribute's Calculation Engine (CE) function, it would not be possible to change that attribute's data element type without first removing the CE function. These types of changes can then be accomplished using the tesserNet scripting tools (rtsh/rtperl).



Verification of the database to ensure database points match the current configuration.

The "verify" functionality is also available externally from the dbBuildCreator tool and can be used by other external applications, including the tesserNet scripting tools.

The dbBuilder tool also provides the ability to run an rtsh script on a matching set of database points. This allows the database maintenance personnel to quickly make database changes to a well defined set of database points.

Security

The dbBuilder, rtsh and rtperl tools follow the standard RTAP database security. The dbBuildCreator tool follows the standard UNIX file system security.

The separation of the dbBuildCreator tool for database design and the dbBuilder tool for database creation allows for the separation of roles between the database designer and the database maintenance personnel.

Summary

This paper has described briefly the typical RTAP database lifecycle and how it can be effectively managed using the tesserNet tools: dbBuildCreator, dbBuilder, rtsh, and rtperl. From point class design through to database maintenance and modifications, the tesserNet tools provide the user with a set of tools that allow them to concentrate on the task at hand.

For more information, please visit the tesserNet web site at `http://www.tessernet.com/` or contact us via email at `info@tessernet.com`.

The dbBuilder tools allows rtsh scripts to be written and run on a well defined set of points.

tesserNet Systems Inc.
142 Sceptre Close N.W.
Calgary, Alberta, Canada
T3L 1Y2
Phone: +1 (403) 237-6835
FAX: +1 (403) 232-8632