

# TPAC Rules Product Description

---

## Overview

TPAC Rules is a rule management system that is designed specifically for the transportation sector. TPAC Rules is provided as a component that can be integrated with third party systems to provide powerful rule management facilities and high-performance rule evaluation.

The TPAC Rules component is comprised of three parts:

- The TPAC Rules Editor, which is used to capture, edit, visualise and test rules
- The TPAC Rules Repository Server, which centrally stores rules and guarantees their integrity
- TPAC Rules application libraries, which allow other applications to apply rules.

The key benefits of TPAC Rules are:

- Rules are controlled by business users - Rule capture and management is straightforward and doesn't require coding
- High-performance rule evaluation, suitable for use for optimisation or real-time validation
- Powerful feature set enables what-if analysis, enterprise-standard version control, accountability for changes and guaranteed rule integrity.

TPAC Rules supports:

- Business processes from planning through to operations and post-operation reporting are supported
- Rules relating to crew, equipment, passengers, freight
- Hard and soft rules, determination of reason messages for illegality or penalties

## Feature Summary

- Rule capture and editing
  - Simple yet powerful approach to rule specification
  - Tailored to handling rules for the transportation sector
- Rule Visualisation
  - View data flow and logical relationship between rules
- Test Environment
  - Build test plans to verify correct operation of rules
- Documentation cross-references
  - Maintain link between documentation
  - Add notes
- Automatic problem analysis
  - Automatically detect inconsistencies in rules within the editor
- What-if analysis
  - Model changes to rules and rule data in isolation
- Version control
  - Rollback functionality

- Accountability for all changes
- Query Revision History
- Security
  - Control who can view and update each rule
  - Group-based security mechanism
- Group collaboration
  - Manage multiple rule repositories
  - Merge changes between versions
- Application Integration
  - C, C++ and Java APIs
  - Flexible interfaces to allow new classes of rules to be specified
- Fast rule evaluation
  - Designed for use within optimisation software and real-time validation systems
  - Provides high-throughput and low-latency rule evaluation
  - Rules are optimised to eliminate redundant checking or calculation
  - C/C++ rule engine compiles rules to machine code for optimal speed

## References

The following documents may also be of interest:

Document	Description
Crewing Rules Overview	A description of how crewing rules are specified for use by products such as TPAC Pairing Optimiser and TPAC Rostering Optimiser.
TPAC Pairing Optimiser Description	An overview of the TPAC Pairing Optimiser component.
TPAC Rostering Optimiser Description	An overview of the TPAC Rostering Optimiser component.
TPAC Rules Editor User Guide	User guide for the editor that is used to specify rules. The guide includes an introduction to TPAC Rules.

## TPAC Rules Editor In Action

TPAC Rules provides a range of features to allow rules to be captured, managed and viewed quickly and easily.

### Rule Capture and Editing

Rules are structured as a hierarchy of simple tables, which can cross-reference one another to implement complex rules. Within the tables, the conditions and outcomes that describe logic can be entered.

The following screenshots show two sample rules that might be applied during the checkin process at an airline's checkin counter. The first rule, named ItemCheck, determines whether an item can be checked in (the IsLegal column in the top right of the screenshot), and if so, whether any surcharge applies (the Surcharge column). The rule table looks like this:

The screenshot shows the Rules Editor interface. On the left is a tree view of the project structure. The main area is split into two panes. The top pane, titled 'ruletable ItemCheck in Checkin/Luggage - Table Editor', contains a table with the following data:

Line Id	Antecedent		Consequents	
	SizeCheck:IsLegal	...	IsLegal [first]	Surcharge [first]
LEGAL	true		true	SizeCheck:Surcharge
ILLEGAL	*		false	0.0

The bottom pane, titled 'ruletable ItemCheck in Checkin/Luggage - Flow Diagram', shows a flow diagram where 'Height', 'Depth', 'PassengerClass', and 'Width' are inputs to the 'Size Limits' rule. The 'Size Limits' rule outputs 'SizeCheck:IsLegal' and 'SizeCheck:Surcharge' to the 'Item Check' rule. The 'Item Check' rule then outputs 'Is Legal' and 'Surcharge'.

The ItemCheck rule table is very simple: it consults the SizeLimits rule to determine if the item is too large, then returns the result. This relationship between the rules is shown in the Flow Diagram view, in the lower half of the screenshot. The ItemCheck can easily be extended to apply other checks in the same way as SizeLimits, such as weight restrictions or aircraft load factors.

The SizeLimits rule inspects the dimensions of the luggage and the class of passenger:

The screenshot shows the Rules Editor interface with the 'ruletable SizeLimits in Checkin/Luggage - Table Editor' pane active. The table contains the following data:

Line Id	PassengerClass	Antecedent				Consequents	
		Depth	Width	Height	Height + Width + Depth ...	IsLegal [first]	Surcharge [first]
STANDARD	*	[0 .. 100]	[0 .. 100]	[0 .. 100]	[0 .. 105]	true	0.0
BUSINESS	'BUSINESS'	[0 .. 100]	[0 .. 120]	[0 .. 100]	[0 .. 115]	true	0.0
FIRST	'FIRST'	*	*	*	[0 .. 130]	true	0.0
OVERSIZE	*	*	*	*	[0 .. 140]	true	25.0
DEFAULT	*	*	*	*	*	false	0.0

The SizeLimits rule table is evaluated from top to bottom, and the results (on the right) are determined by the first matching line.

By linking simple rule tables together, complex rules can be represented and managed easily.

## Documentation Cross-References

The TPAC Rules Editor allows rule maintainers to add references to rule lines, rule tables and sets of rule tables. These can include:

- Hyperlinks, perhaps to statutory documentation on the web
- References to local files

- Plain text
- Boolean flags

Using this mechanism, the source of each rule can be made clear, and any additional notes can be added as plain text. By maintaining this coupling between rule logic and documentation, it is easy both to keep rules in sync and to verify that the rules are in sync.

## Test Environment

The TPAC Rules Editor provides a built-in test environment, allowing users to try out rules as they are being built. The user can specify any number of test cases, each of which provides input data for the test and the expected outcomes. The rules can then be applied, and actual results are compared with expected results.

Tests can be saved as test plans, which can be used to ensure that rule changes don't unintentionally cause other rules to break.

The test plan below shows five test cases for the ItemCheck rule:

The screenshot shows the Rules Editor interface. The top part displays a rule table for 'ruletable ItemCheck in Checkin/Luggage - Table Editor'. The bottom part displays a test environment table for 'ruletable ItemCheck in Checkin/Luggage - Test Environment' with a test plan named 'ItemCheckTest'.

Line Id	Antecedent		Consequents	
	SizeCheck:IsLegal	Additional Conditions	IsLegal [first]	Surcharge [first]
LEGAL	true		true	SizeCheck:Surcharge
ILLEGAL	*		false	0.0

Test Plan: ItemCheckTest									
Case Id	Input Attributes				Expected Results		Actual Results		
	PassengerClass	Height	Width	Depth	islegal	surcharge	islegal	surcharge	
Surcharge	'BUSINESS'	60	40	35	true	25.0	true	25.0	✓
IllegalEconomyWeight	'ECONOMY'	35	50	20	false	0.0	true	0.0	✗
IllegalEconomySize	'ECONOMY'	60	50	30	false	0.0	true	25.0	✗
LegalFirst	'FIRST'	40	50	30	true	0.0	true	0.0	✓
LegalEconomy	'ECONOMY'	35	50	20	true	0.0	true	0.0	✓

In this example, two test cases have failed, which indicates that either the rules or the test cases need some attention.

## What-If Analysis

TPAC Rules is designed to allow what-if scenario modelling, so that changes to rules or reference data can be made in a non-production environment to assess the impact of those changes.

A common usage of this capability is to model changes to crew rules during enterprise bargaining, and passed to crewing optimisation software to model the impact on staff levels, pay costs and operational capacity.

The way this typically works is to clone the production repository and make changes. Applications that use TPAC Rules, such as optimisers, or equipment management systems can then be directed to use the non-production repository as their source of rules. The Revision History feature can be used at any to see what has changed since the clone from production, and the Merge feature allows the changes to be merged back into production if desired.

This modelling capability possible due to TPAC Rules repositories being entirely self-contained, which allows applications to retrieve all rule information from one central location. Any reference data that is owned by external systems but required by the rules is internalised into the rule repository.

## Accountability

For every change that is made to rule repositories, the TPAC Rules Repository Server stores:

- The name of the user that made the change
- What was changed
- A change description, entered by the user.

This information can be queried from the TPAC Rules Editor to find the change history of any rule table, set of rule tables or the whole repository.

## Group Collaboration

Multiple users can view and edit the same rule repository, then use the TPAC Rules Editor's merge facility to combine any changes.

During a merge, all changes are summarised with details of who made the change. Each change may be applied, and if any conflicts occur (such as two edits to the same cell in a rule table), the user can decide which change to apply, and what other edits may be required as a result. Any tests that may have been broken can be repaired, and the TPAC Rules editor automatically analyses the repository for any logical errors.

## Rule Migration

The TPAC Rules Repository Server can manage multiple rule repositories, and provides the facility to clone (or branch) repositories, work on them, then merge any changes back in to the parent repository.

As an example, a rule maintainer in the crewing department might create a clone of the PRODUCTION rule repository in January, in order to begin working on some crewing rule changes. During February, a rule maintainer in the Operations department might modify some equipment loading rules in the PRODUCTION repository. In March, the crewing rule changes are done. Using the TPAC Rules Editor's merge facility, the changes to the crewing rules can be incorporated into the PRODUCTION repository, and any conflicts with the changed equipment loading rules can be detected and resolved.

Although that example describes rule maintainers making changes directly to the PRODUCTION repository, it is more common to set up a STAGE or DEVELOPMENT repository to which all rule changes are made. Then, one authorised party can verify that all changes are correct before migrating the changes to the PRODUCTION repository. The TPAC Rules security model enables this practice to be enforced.

## Performance

Within the transportation sector, rules need to be applied fast. Rules need to be applied in real time to detect operational problems, to validate users' actions in scheduling and operations control software, and to guide planning or disruption recovery optimisers to solutions quickly.

TPAC Rules was designed specifically to provide the required performance for the classes of rules that are common in the transportation sector.

This is achieved through:

- Static redundancy analysis to eliminate repeated calculations.
- Compilation of rules to machine code for optimal execution speed.
- Techniques to avoid rules being reapplied if nothing has changed.
- Dynamic execution reordering to capitalise on some rules firing more often than others without requiring tuning.

Using these techniques, TPAC Rules has been shown to have the performance required for both real-time front-end applications, and optimisation software. TPAC Rules is integrated with the TPAC Pairing Optimiser, and the TPAC Rostering Optimiser, both of which demand exceptional rule evaluation performance.

## Technical Overview

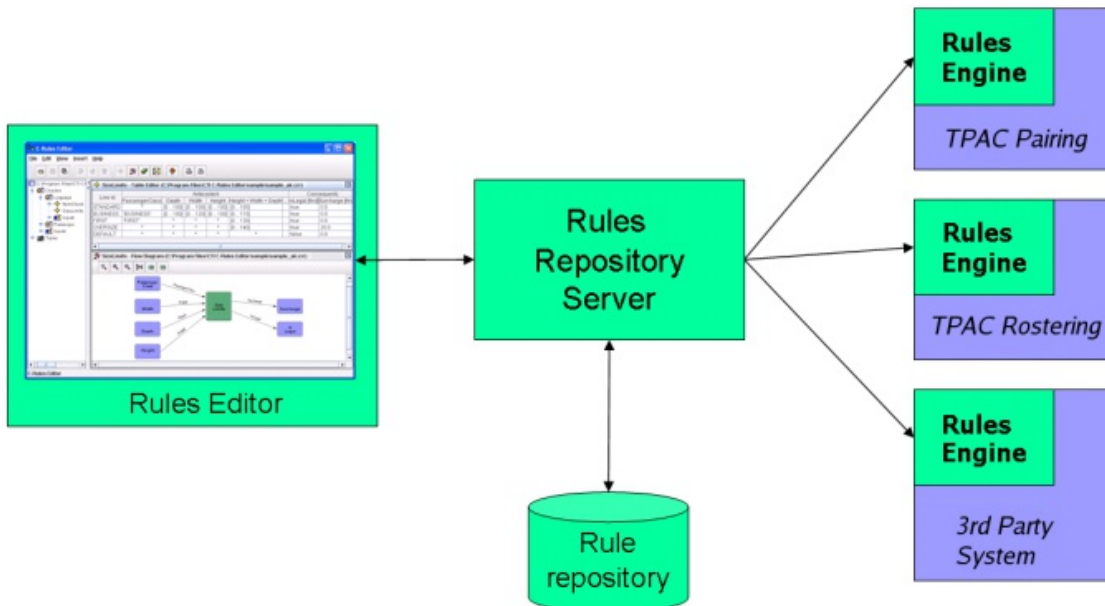
## Architecture

The following diagram describes the architecture of the TPAC Rules component. It shows:

- The TPAC Rules Repository Server, which is the central rule store. It runs within J2EE application server, and is responsible for serving rules to the TPAC Rules Editor and TPAC Rules Engines and maintaining their integrity.
- The TPAC Rules Editor, used by rule maintainers. The editor connects to the TPAC Rules Repository Server using HTTP-based messages to retrieve and save rule repositories. It can also be used in a standalone mode, in which case it can load and save XML file-based repositories.
- Applications that apply rules. Such applications are linked with a TPAC Rules Engine, which is provided as a library. Two sample applications, the TPAC Pairing Optimiser and the TPAC Rostering Optimiser are shown in the diagram.

The library is capable of connecting to a TPAC Rules Repository Server, retrieving rules, compiling them in preparation for use, and applying rules as required by the application. Also, any changes to the rules made on the Repository Server can be detected and applied if desired by the application.

The TPAC Rules Engine can be incorporated with native applications, Java applications, PL/SQL programs running in Oracle Databases, or any other application hosts that can access libraries either as native shared objects or as Java class libraries.



## Supported Platforms

### TPAC Rules Editor

**Table 1. TPAC Rules Editor Supported Platforms**

Type of application	Java application
Required hardware	PC
Operating System	Windows, Linux, Solaris or any other platform that supports the Java 5 runtime environment
Memory	512MB minimum
Disk space	30MB for full installation

### TPAC Rules Repository Server

**Table 2. TPAC Rules Repository Server Supported Platforms**

Type of application	Java J2EE application
Required hardware	Server
Operating System	Windows, Linux, Solaris or any other platform that supports the Java 1.4 runtime environment
Application Server	Oracle Application Server (proprietary) or Tomcat Application Server (open source)
Memory	512MB minimum
Disk space	30MB for full installation

### TPAC Rules C/C++ Rule Engine (libcrule)

**Table 3. TPAC Rules C/C++ Rule Engine Supported Platforms**

Type of application	Native, written in C/C++
Required hardware	PC or Server with Intel-compatible or SPARC CPU
Operating System	Linux (RHEL4) or Solaris 10, ports to other platforms available on request
Memory	256MB minimum
Disk space	60MB for full installation

### TPAC Rules Java Rule Engine

**Table 4. TPAC Rules Java Rule Engine Supported Platforms**

Type of application	Java class library
Required hardware	PC or Server
Operating System	Windows, Linux, Solaris or any other platform that supports the Java 1.4 runtime environment
Memory	128MB minimum
Disk space	30MB for full installation

## APIs for Application Integration

Three APIs are available for application integration. This range of APIs allows the TPAC Rules rule engine to be integrated with front-end applications for local validation, server software for back-end rule application, and backend software such as optimisers.

The choice of API to use for integration with an application will largely be determined by the language in which the application is written.

- A C API, which exposes a range of initialisation, data binding and rule execution functions. The functions are described by ANSI C header files, and implemented by the libcrule library.

The library is provided as a shared object file which can be loaded and run by native applications, application containers such as Oracle Database (using the PL/SQL External Procedures interface), and Java applications (using JNI, though the Java API would usually be the preferred option).

- A C++ API, which exposes a similar functionality to the C API, but uses the C++ language to provide the functionality at a higher level. The C++ API is provided entirely as header files that wrap around the C API, so the library is not subject to C++-compiler ABI incompatibilities.
- A Java API, enabling Java applications to apply rules. The Java Rule Engine is written in 100% Java, enabling the library to run in any environment that supports the Java 1.4 or Java 5 Runtime Environment.

Please contact Constraint Technologies for further information about the TPAC Rules Rule Engine

APIs.

## Further Information

You may wish to look at the [PDF version](#) of this document.