GRS Best Practice Guide

New Features by Release

4/1/2026

New Features by Release

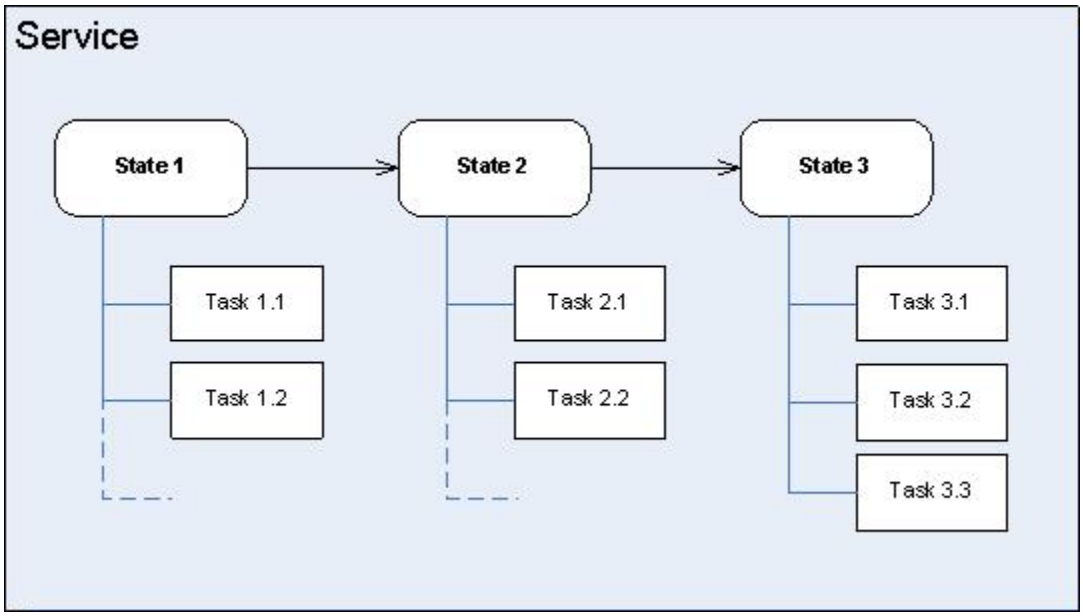
New in 8.5.1

Support for Conversation Manager Templates in Test Scenarios

In the initial 8.5.001 release of GRS, the Test Scenario feature did not support rules that were created using the Conversation Manager (CM) template. This is because the Test Scenario feature in release 8.5.001 works by taking the input data (a set of one or more facts with different fields) that is configured by the user and building the appropriate Fact model, then running the rules under GRAT using that set of data. In release 8.5.1, the Test Scenario feature now supports rules based on the CM template.

Data Structure in CM

With Conversation Manager, the data is in a hierarchical JSON format of **Customer -> Service -> State -> Task**. Any given **Customer** may have one or more **Services**. Each **Service** may be in at most one **State** at a time. Each **State** may have one or more **Tasks**. **Tasks** may also be associated directly with **Services**.



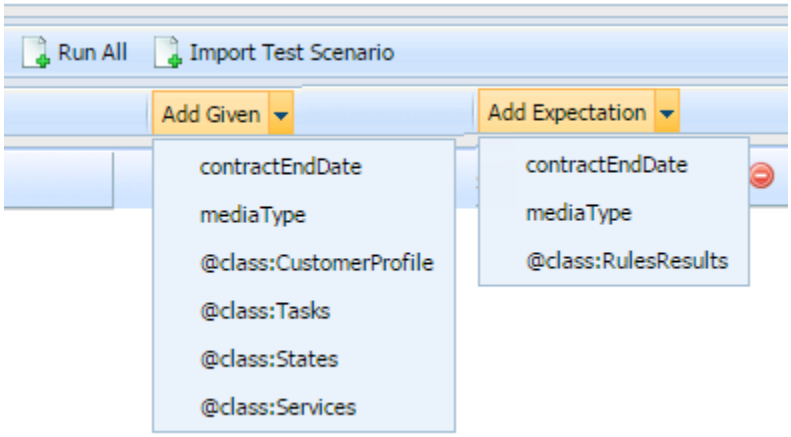
So the `Customer`, `Services`, `States` and `Tasks` Facts have now been added the lists of Facts that can be defined as **Given** fields, and the `RulesResults` Fact has been added to the list of Facts that can

be defined as an **Expectation**.

> **Important**
>
> The current CM Template is only interested in the Type, Start Time, and Completion Time (if any) of Services, States, and Tasks.

Each of the new values is represented by a JSON string which will be the value for that field.

Now, when the type of rule for which you want to create a test scenario is a Conversation Manager rule (based on the Conversation Manager template), a series of different values for the **Given** and **Expectation** elements that reflect these more complex data structures are available. In the example below you can see the **Customer > Service > State > Task** structure is reflected by the four **@class** entries in the drop-down list of Givens and the **@class:RulesResults** entry in the drop-down list of Expectations.



When you select an **@class** entry, a new column is added. Click on a grid cell under the new column to bring up the edit dialog for that entry. The additional data listed below can be selected as either a **Given** or an **Expectation**.

Additional CM Template Objects

Givens

The list below shows the additional provided data.

- Available by selecting one of the **@class** entries:
  - `Add Customer Attribute`
  - `Add Service`

    - Add Service Type
    - Add Service Start Time
    - Add Service Completion Time
    - Add State
    - Add State Type
    - Add State Start Time
    - Add State Completion Time
    - Add Task
    - Add Task Type
    - Add Task Start Time
    - Add Task Completion Time
- Available for direct selection from **Givens**:
    - Add Interaction Media Type
    - Add Contract End Date

Expectations

The list below shows the additional expected results:

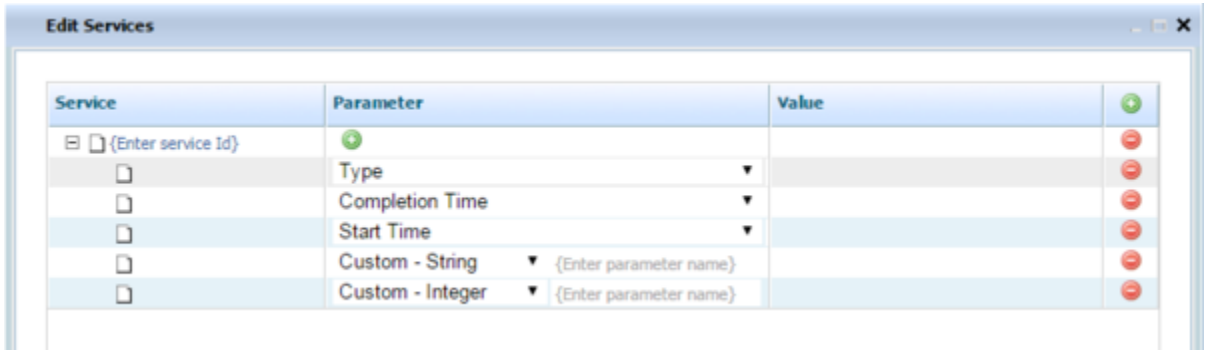
- Update Customer Attribute
- Request Specific Agent
- Request Agent Group
- Request Place Group
- Request Skill
- Send Communication to Customer
- Block Communication to Customer
- Offer Service Resumption
- Offer Survey to Customer

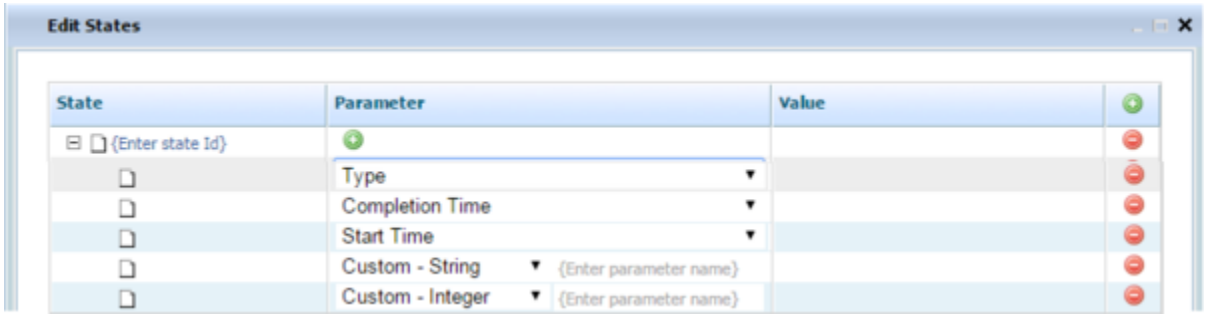
Edit Dialogs

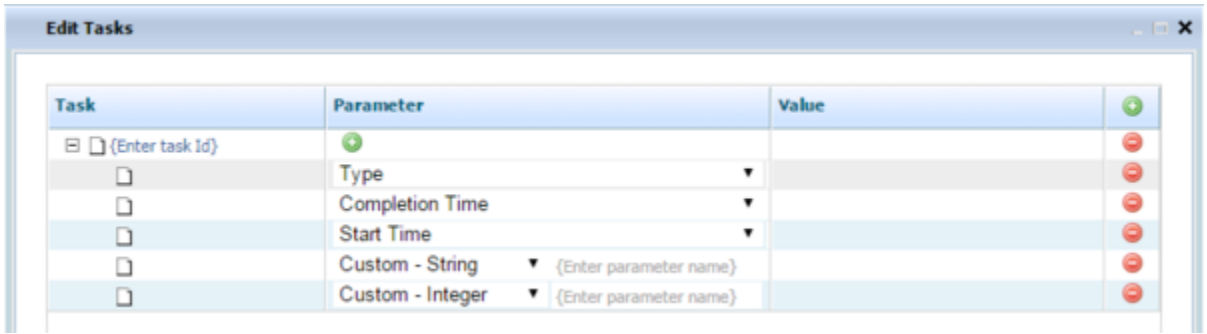
To create entries for the **Givens** and Expectations of your Conversation Manager test scenario, select the relevant **@class** item and use the sample additional edit dialogs shown below.

Givens

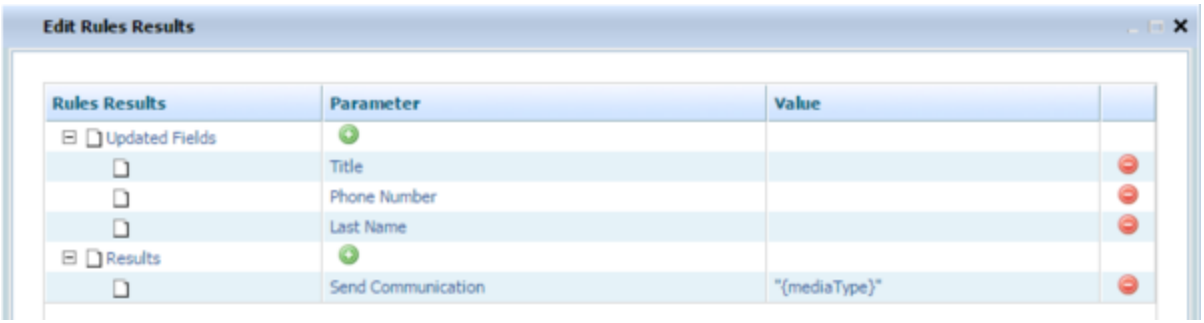

Edit Customer Profile
Customer
Parameter
Value
Customer
Title
Phone Number
Last Name


Edit Services
Service
Parameter
Value
{Enter service Id}
Type
Completion Time
Start Time
Custom - String
{Enter parameter name}
Custom - Integer
{Enter parameter name}


Edit States
State
Parameter
Value
{Enter state Id}
Type
Completion Time
Start Time
Custom - String
{Enter parameter name}
Custom - Integer
{Enter parameter name}


Edit Tasks
Task
Parameter
Value
{Enter task Id}
Type
Completion Time
Start Time
Custom - String
{Enter parameter name}
Custom - Integer
{Enter parameter name}

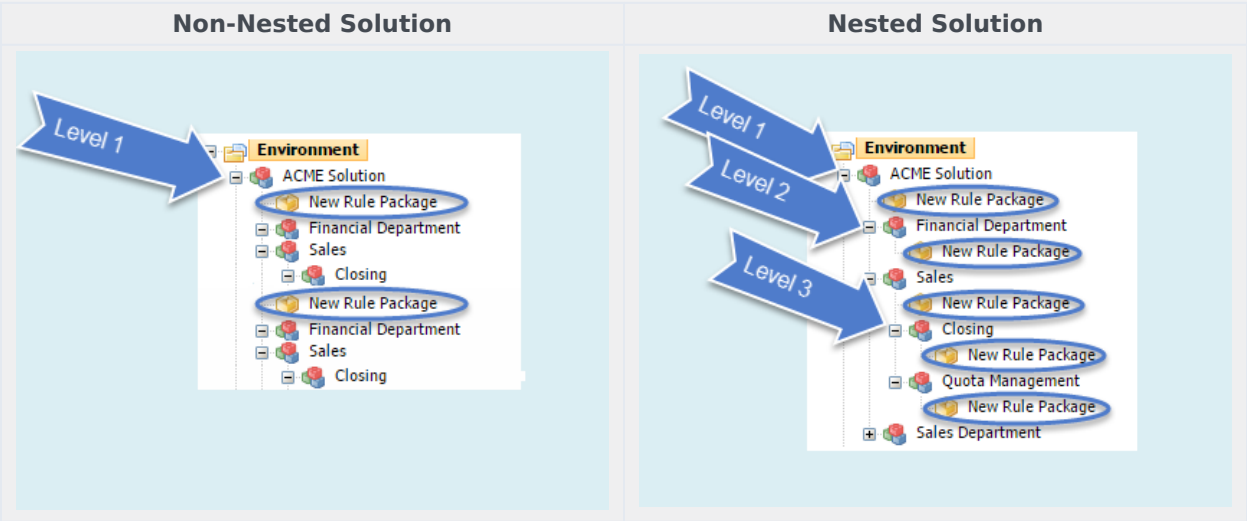
Expectations



Nested Solution Business Hierarchy

In release 8.5.1 of Genesys Rules Authoring tool, if you have permission to create a new rule (`Rule Package - Create`) you can now add a new Rule Package at any node in the business hierarchy (a *nested* solution), rather than just at the first level.

[+] FULL DESCRIPTION



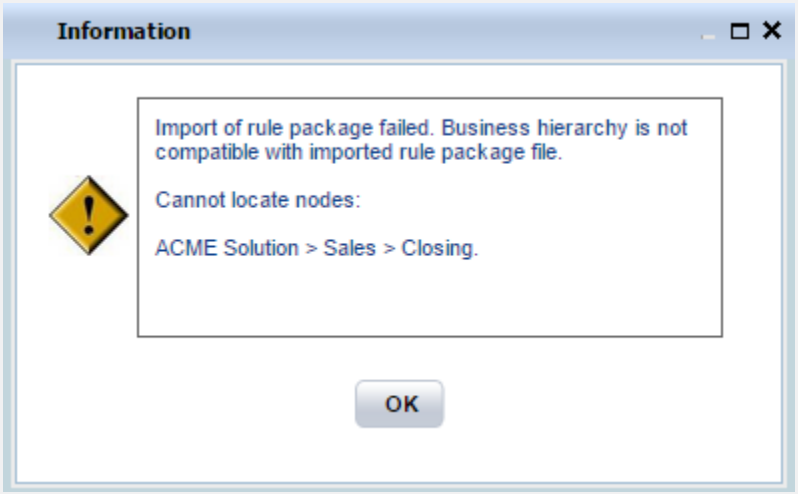
This means that:

- Your business hierarchy can be more easily organized.
- The need for lots of duplication and repetition at the Solution level in more complex business hierarchies is now removed.
- Individual users can be restricted using Role-Based Access Control to specific sub-nodes (for

example, Departments and Processes).

Importing Rule Packages

Because rules can be associated with sub-nodes in a nested hierarchy, when a rule package is imported, GRAT ensures that the business structure is compatible, and prevents an import if it is not. If GRAT finds an incompatibility, an error such as the following is displayed:

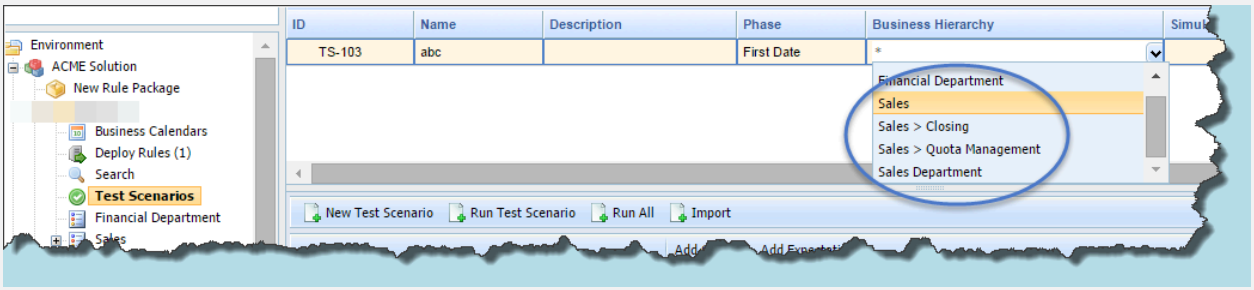


Important

Even if the **Auto-create business hierarchy during import** button is selected, GRAT prevents the same node name from being created anywhere in the hierarchy—uniqueness of business node names across the entire hierarchy is still enforced.

Test Scenarios

For Test Scenarios, the Business Hierarchy drop-down displays the relative path underneath the selected rule package:



Deleting Business Nodes

Be careful never to delete any business structure nodes that contain active rule packages or rules, without first backing up the rule packages as XML files. While it is OK to add new nodes, or to "rename" nodes, proper permissions should be set up to prevent a GA/GAX user from accidentally deleting nodes which could cause rule packages / rules to become unreachable.

Enabling and Disabling the Feature

Because some customers might want to restrict their users to creating rule packages only under the **Solution** node, in release 8.5.100.21 a new configuration option—`enable-nested-solutions`—has been implemented to allow users to enable or disable this feature. Disabling this feature is recommended for iWD users.

- Option name—`enable-nested-solutions`
- Valid values—`true/false`
- Default value—`true`
- Description—Controls whether users can create new rule packages under any node in the hierarchy. For iWD, it is recommended to set this option to `false`.

New in 8.5.001.21

Business Calendar Enhancements

Business calendars have been enhanced in GRS release 8.5.2 to allow:

- Dynamic Timezone Support
- Differentiation between holidays and non-working days.

[+] FULL DESCRIPTION

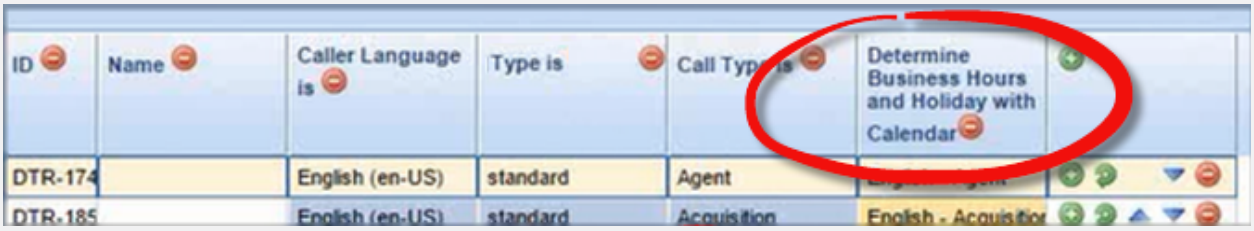
Dynamic Timezone Support

When the GRAT user configures a business calendar, a timezone is chosen along with the other attributes of the calendar (normal work week, exceptions, holidays). Business calendars have been enhanced to allow the timezone to be provided dynamically at rule-evaluation time. This feature enables you to configure a standard set of business calendars that can be re-used in any timezone.

In this release, the standard methods that can be accessed from within the rule template have been extended to allow the timezone ID to be passed in at rule evaluation time. If the timezone ID is not passed in in this way, then the "saved" timezone is used. If the timezone ID is passed in, then it overrides the saved timezone and the calculations will be done using the provided timezone.

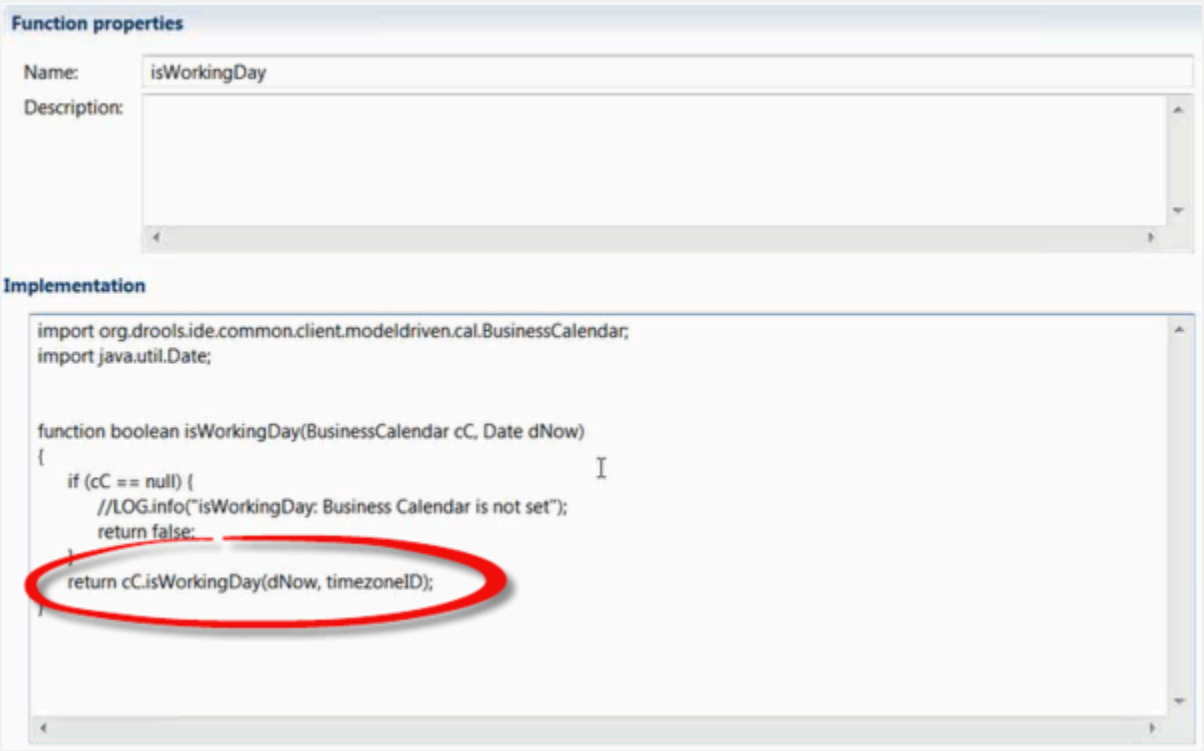
Example

A rule, such as in the example below, asks which business calendar to use in evaluating the rule conditions;



and this can answer questions about whether today is a working day (by using the `isWorkingDay` method), or whether this time is in business hours, and so on. Now, you can configure the calling application (that is calling for the rule evaluation) to pass in a timezone ID as a parameter to the relevant business calendar function, and if present, this will override the configured timezone of the business calendar that the rule was created with.

For example, for the `isWorkingDay` method, the addition of the `timezoneID` parameter (highlighted below in the GRST template) enables this override to occur if the parameter is passed in:



The `isWorkingDay` condition is then evaluated based on the dynamic timezone passed in by the calling application.

New Method Signatures to the BusinessCalendar Object

The following new method signatures have been added to the `BusinessCalendar` object, and can be invoked from within a rule function:

- `public boolean isWorkingDay(Date theDate, String timeZoneID);`
- `public boolean isHoliday(Date theDate, String timeZoneID);`
- `public boolean isException(Date theDate, String timeZoneID);`
- `public boolean isWorkingTime(Date theTime, String timeZoneID);`
- `public int diffWorkingDays(Date date1, Date date2, String timeZoneID);`
- `public int diffWorkingHours(Date date1, Date date2,String timeZoneID);`
- `public int diffWorkingMinutes(Date date1, Date date2, String timeZoneID);`
- `public long diffWorkingSeconds(Date time1, Date time2, String timeZoneID);`

- `public Date beginningOfWorkingDay (Date time, String timeZoneID);`
- `public Date endOfWorkingDay (Date time, String timeZoneID);`

Differentiation between Holidays and Non-Working Days

Business calendars have been enhanced to distinguish between holidays and non-working days. Four new methods have been added to the business calendar object:

- `isHoliday()`—Returns whether this calendar day is a holiday. Holidays are always non-working days.
- `isHoliday(date)`—As for isHoliday() but allows you to specify the date to check.
- `isException()`—Returns whether the day is an exception to the standard work schedule. An exception includes either a time change or a holiday.
- `isException(date)`—As for isException() but for a specified day.