Time in ServiceNow

Introduction to Time-Related Functionality

Introduction to Time-Related Functionality

Overview

There are a number of different functionalities that allow for tracking time and leveraging that information across applications.

Time Fields

Time can be stored in different ways in records. Understanding how time is stored underlies use of all of the time functions.

For more information, see: Using Date and Time Fields.

Time Zones

All times are stored in the platform in Universal Coordinated Time. They are displayed globally based on the system time zone, but are displayed to users in their local time zone according to user settings. Time zone information is important to keep track of to avoid calculation errors or confusing outputs.

All dates and times gathered through web services display in GMT. Data inserted using a web service uses the active user's time zone or the system time zone if the active user does not have a time zone specified.

For more information, see Using Time Zones.

Schedules

Schedules are rules which include or exclude ranges of time for certain time-related functionality. For example, schedules can restrict SLAs to only count time during business hours.

For more information, see Using Schedules.

Displaying Time

There are a number of useful user interfaces that help represent time visually.

For more information, see Displaying Time.

Viewing Logs

Logs within the system provide historical information as to what occurred in the instance when.

For more information, see Viewing System Logs.

Scheduling Events

These functionalities allow other functionality to be triggered at particular times, or in response to specific events.

For more information, see Scheduling Events.

Timing Functionality

These functions gather information with regards to duration. They answer the question "How long?", and can time events based off of that information.

For more information, see Timing Functionality.

<!--

Calculating Time

These functionalities allow for the manipulation of time information.

For more information, see Calculating Time.

Using Date/Time Fields

Note: *This article applies to Fuji. For more current information, see Date and Time Fields* [1] *at* http://docs.servicenow.com The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

Records can store date and time values in several different types of fields. These values are stored in the database as integer numbers of milliseconds, and are displayed in the appropriate date or time format.

Date and Time Field Types

The following field types are provided for storing date and time information in records.

Field Type	Dictionary XML Type	MySQL DB Type
Date	glide_date	DATE
Date-Time	glide_date_time	DATETIME
Time	glide_time	DATETIME
Duration	glide_duration	DATETIME
Due-Date	due_date	DATETIME

For the full list of field types, see Introduction to Fields.

Formatting Date and Time Fields

When modifying the default date and time formats, also verify the format using a Validate Date and Time script.

Date Format

The date format is defined by the property `glide.sys.date_format`. An administrator can modify the property by navigating to **System Properties > System**. Use the same 'format' strings as the `java.text.SimpleDateFormat` class, with minor exceptions. Note that MM is months, where mm indicates minutes. The format string consists of the following abbreviations.

Field	Full Form	Short Form
Year	yyyy (4 digits)	yy (2 digits), y (2 or 4 digits)
Month	MMM (name or abbr.)	MM (2 digits), M (1 or 2 digits)
Day of Month	dd (2 digits)	d (1 or 2 digits)

The default format is: **yyyy-MM-dd**.

Time Format

The time format is defined by the property `glide.sys.time_format`. An administrator can modify the property by navigating to **System Properties > System**. Use the same 'format' strings as the `java.text.SimpleDateFormat` class, with minor exceptions. The format string consists of the following abbreviations.

Field	Full Form	Short Form
Hour (1-12)	hh (2 digits)	h (1 or 2 digits)
Hour (0-23)	HH (2 digits)	H (1 or 2 digits)
Minute	mm (2 digits)	m (1 or 2 digits)
Second	ss (2 digits)	s (1 or 2 digits)

The default format is: **HH:mm:ss**.

User Preferences

Users can personalize the format in which date and time values appear in the instance. Personalizing the date or time format does not change global settings or impact the way other users see date values.

1. Navigate to **Self-Service > My Profile**.
2. Select options for the **Date format** and **Time format** fields.

   Note: An administrator must add the **Time format** field to the Self-Service view of the User form. For more information, see Configuring Forms.
3. Click **Update**.

Configuring the Date Picker for the List Editor

In UI15, a system property enables you to choose between two date picker configurations for the list editor (starting with the Fuji Patch 5 release)

1. Navigate to **sys_properties.list**.
2. Search for the property named `glide.ui.list_edit.show_calendar_only`.
3. Set the property **Value** to either of the following options.
   - **false:** The date picker displays a calendar as well as a field for manual date entry. This is the default behavior in UI11, regardless of the property value.
   - **true:** The date picker displays a calendar only. This is the default behavior in UI15.

Default Date and Time Fields

Certain time fields are provided by default to store particular date and time fields.

Global Timestamp Fields

All records inherit the following time stamp fields from the Global [global] table:

- **Created**
- **Updated**

These fields are automatically populated with the correct date and time.

Task Fields for Measuring Work Time

The following base system fields are provided on certain tables for keeping track of how long it takes to close tickets:

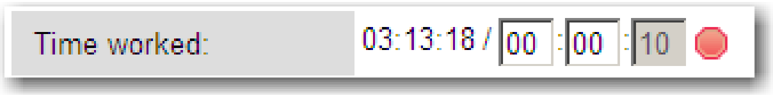
- **Time worked**: A timer which runs while the record is being viewed by a user, and pauses while the record is closed (or when it is paused manually). Used to keep track of the time spent by the help desk while working on the record.
- **Resolve time**: A calculated field which measures the time from the moment the record is opened, to the moment the record is closed. Used to keep track of how long it takes to resolve the record.

These fields provide different metrics for request response.

The following additional tools are available for tracking work time:

- **Service level agreements (SLAs)**: measure how long it takes a record fulfill certain conditions (such as an incident being marked **Resolved**).
- **Time cards**: use the **Time worked** field to break down how much time was spent by day of the week.

Time Worked



The Task [task] table provides a time-tracking field called **Time worked**. This field measures how long a record has been viewed in order to measure work time on a ticket. Any table that extends Task can use this field. To add the field, simply personalize the form.

As the record is viewed, the timer counts upward. To pause the timer, click the stop icon (🔴); to resume the timer, click the start icon (▶).

When the task is saved, the amount of new time in the timer is used to generate a record on the Time Worked [task_time_worked] table. This table can be viewed as a related list on the task form.

By default, the time displayed in the **Time worked** field displays a cumulative value stored in the task record. If you modify a Time Worked record, the changes will not be reflected in the task timer.

You can set the property `com.snc.time_worked.update_task_timer` to enable updating of the task timer value based on changes to the time worked records. This is accomplished through the **Update task timer** business rule.

Resolve Time

The **Resolve time** field is available on the Incident [incident] and Request [sc_request] tables. The field is calculated by business rules when the record is marked closed, and measures the difference between the **Opened by** and **Closed by** dates. This field allows for easy reporting on how long it takes for requests to be closed.

The field is stored in the system as an integer number of seconds.

Displaying Resolve Time as a Duration

To display the resolve time as a human-readable duration rather than an integer number of seconds:

1. Right-click the field on the form.
2. Select **Configure Dictionary** (**Personalize Dictionary** in versions prior to Fuji).
3. Enter **format=glide_duration** in the **Attributes** field. If attributes are already there, add the new attribute separated by a comma and no space.
4. Click **Submit**.

   The resolve time will now display in a number of days, hours, and minutes.

Note: *This change affects only the display on forms and lists. It does not change the display for reports. Consider reporting duration rather than resolve time to see hours and minutes in a report.*

Business Rule Calculation

On the Incident table, the field is calculated on closure by the business rule **mark_closed**. The following lines of code calculate the resolve time:

```
current.calendar_stc =
  gs.dateDiff(
     current.opened_at.getDisplayValue(),
     current.closed_at.getDisplayValue(),
     true);
```

On the Request table, the field is calculated on closure by the business rule **Mark Request Closed**. The following line of code calculates the resolve time:

```
  current.calendar_stc =
gs.dateDiff(current.opened_at.getDisplayValue(),current.closed_at.getDisplayValue(),true);
```

Planned Task Time Fields

The Planned Task Plugin provides a table (Planned Task [planned task]) with standard fields for measuring a planned task's time. For more information, see Planned Task.

Exporting Date and Time Information

Because some export formats are intended for human consumption and others are intended for database usage, different methods provide date and time field information in different formats.

Excel

Date, Date-Time, and Time fields are all exported as their display values, displayed using a custom format instead of the system date format.

Duration fields, however, export as the value stored in the database, which is an integer value of seconds.

XML

All Date and Time fields export as the value stored in the database.

PDF

All Date and Time fields (including Duration) export as their display value.

CSV

All Date and Time fields export as the value stored in the database.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/reference/r_UseDateAndTimeFields.html

Using Time Zones

Note: *This article applies to Fuji. For more current information, see Time Zones* [1] *at* http://docs.servicenow.com The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

All times are stored in the platform in Coordinated Universal Time (UTC). They are displayed globally based on the system time zone, but are displayed to users in their local time zone according to user preferences.

Time zone representation

Time zones that have the Country/City format are primary time zone IDs. Other time zone IDs are links to the primary time zone. For example US/Pacific is a link to the America/Los_Angeles time zone. Both America/Los_Angeles and US/Pacific represent Pacific Standard Time with the same zone offset and Daylight Savings Time (DST) schedule.

Other than the representation, there is no impact on date and time functionality.

In the absence of a default time zone for the user or the system, JVM read's default time zone information from the machine, and depending on how the machine is configured, it might return the Country/City or link, for example, US/Pacific or America/Los_Angeles. It is recommended that administrators configure their system with a default timezone "glide.sys.default.tz" to avoid system dependencies.

Daylight Savings Time

If a time zone is specified based on location (for example, **America/Los Angeles**), times will automatically be adjusted for daylight savings time. If a time zone is specified based on the name of a time zone (for example, **GMT**), it will not adjust.

System Time Zone

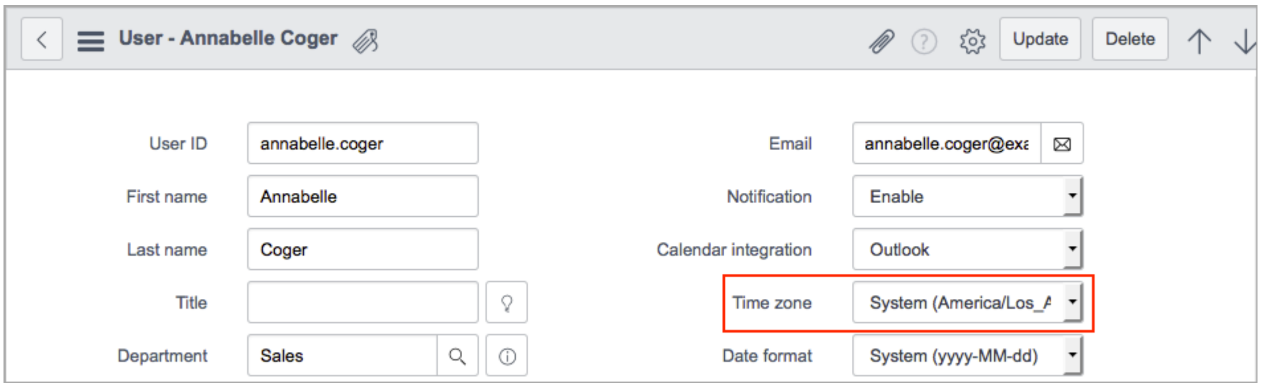
Setting System Time Zone

To set a default time zone for calendars and users:

1. Navigate to **System Properties > System**.
2. Locate the property called **System timezone, used as default for calendars and users.**

   By default, the input field is blank. If no time zone is defined for this property, **America/Los Angeles** is used as the default.
3. Type a time zone in the field and click **Save**. This becomes the system time zone.

   The new time zone automatically cascades to all users who do not already have a specified time zone. If a user selects a different time zone or if the administrator selects a different time zone for them, the user is assigned the selected time zone and does not use the system time zone anymore.



User - Annabelle Coger

Update Delete

User ID	annabelle.coger	Email	annabelle.coger@exa
First name	Annabelle	Notification	Enable
Last name	Coger	Calendar integration	Outlook
Title		Time zone	System (America/Los_A
Department	Sales	Date format	System (yyyy-MM-dd)

User Preferences

Once the System Time Zone is defined, users can also select their own time zone from their user form, accessed through **Self-Service > My Profile**.

The System default will always be displayed as **System ([name of the default time zone])**. For example, if the System time zone is **America/Los_Angeles**, the user will see **System (America/Los Angeles)**.

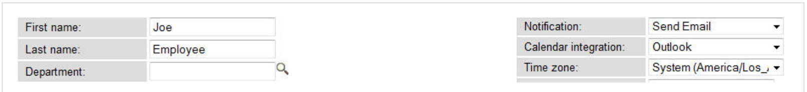
Time Zone Choice List

Wherever users have a choice of time zone, the choices are populated using the **Time Zone** choice list on the **User [sys_user]** table. Not all time zones appear by default.

Warning: These instructions refer to making existing, base-system options visible in the choice list. Do not create new values for this field in the dictionary.

To add or remove time zones from the list of time zones:

1. Navigate to **User Administration > Users** and open any user record, or click **New**.

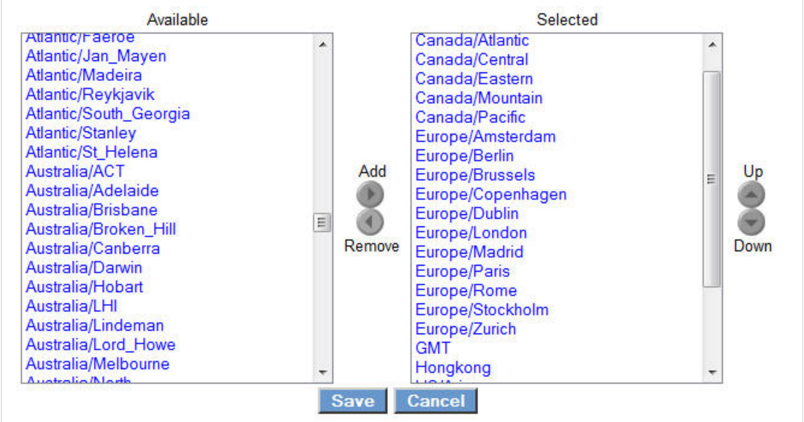


First name:	Joe	Notification:	Send Email
Last name:	Employee	Calendar integration:	Outlook
Department:		Time zone:	System (America/Los_

User record

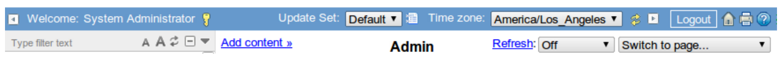
Notice the default time zone is **System (America/Los_Angeles)**.

2. Right-click **Time zone**, and select **Configure Choices** (**Personalize Choices** in versions prior to Fuji).
3. Highlight the desired time zone from the **Available** or **Selected** lists, and then **Add** or **Remove** the time zones as needed.



Choosing time zones

The Time Zone Changer



The time zone changer

The Time Zone Changer allows users to temporarily change their time zone for a session, using a drop-down selector in the tool bar. By default, the Time Zone Changer is inactive.

A temporary time zone changer can be used by:

- Users who travel frequently between timezones
- Service desk employees who want to work in the same time zone as the users they are helping
- Administrators who are designing Schedules and SLAs

To activate the Time Zone Changer for the entire instance:

1. Navigate to **System UI > UI Macros** and open the **ui_timezone_changer** macro's form.
2. Check the **Active** field.
3. Refresh the browser.

The Time Zone Changer will now be visible in the browser for administrators. To change what roles have access to the Time Zone Changer, navigate to **sys_properties.list** and edit the record **glide.timezone_changer.roles**. The property accepts a comma-separated list of roles.

Time Zones in Scheduled Reports

By default, Scheduled Reports are generated in the time zone of the user who runs them (the user in the **Run As** field). However, there is a way to manually specify the time zone for the report.

To change the time zone in which the scheduled report runs:

1. Configure the Scheduled Report form and add the **Run As tz** field.
2. Select the appropriate time zone.

Time Zones in Scheduled Data Imports

By default, scheduled data imports are run using the time zone of the user who creates them. However, there is a way to manually specify the time zone for the import.

To change the time zone of the scheduled data import:

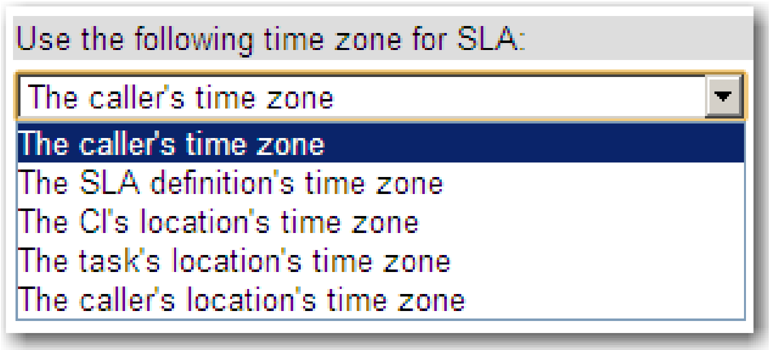
1. Configure the Scheduled Data Import form and add the **Run As tz** field.
2. Select the appropriate time zone.
3. Click **Update**.

Time Zones in Email Notifications

The date and time stamp of a notification uses the system time zone and not the time zone of any recipient. The property `glide.email.append.timezone` in **System Properties > Email** controls whether to append the time zone. If true, the system time zone of the instance is appended to any dates or date/times in outbound email messages (for example, 2010-07-02 04:01:14 PST).

Time Zones in Service Level Agreements

Service Level Agreements have a number of options of which time zone to use. To set a time-zone for SLAs, navigate to **Service Level Management > SLA Properties** and locate the following property:


Use the following time zone for SLA:
The caller's time zone
The caller's time zone
The SLA definition's time zone
The CI's location's time zone
The task's location's time zone
The caller's location's time zone

Time zone for SLA

Some special considerations:

- If **The caller's time zone** is selected, there will be unpredictable behavior if the caller does not have a time zone defined.
- If **The SLA definition's time zone** is selected, the time zone must be manually defined on the SLA's form.

For more information on defining an SLA, see Defining an SLA with Plugin.

Time Zone in Scripting

When scripting on the server, there are a number of GlideSystem Date and Time Functions used to get time values. Consult the API for information about specific methods and to learn the format in which each returns the requested time.

Enhancements

Dublin

- DST enhancement requires that reports and queries observe daylight saving time rules. The changes affect trend charts, line charts, and filters using the "trend on" operation.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/reference/r_TimeZones.html

Using Schedules

Note: *This article applies to Fuji and earlier releases. For more current information, see Schedules* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

Schedules are rules that include or exclude time for various actions or tasks. Use schedules to specify when service level agreements or inactivity monitors are active, or to specify when on-call rotations should take effect. For example, if a service level agreement is set to an 8-5 Weekdays schedule, the SLA only counts time during those hours.

Defining Schedules

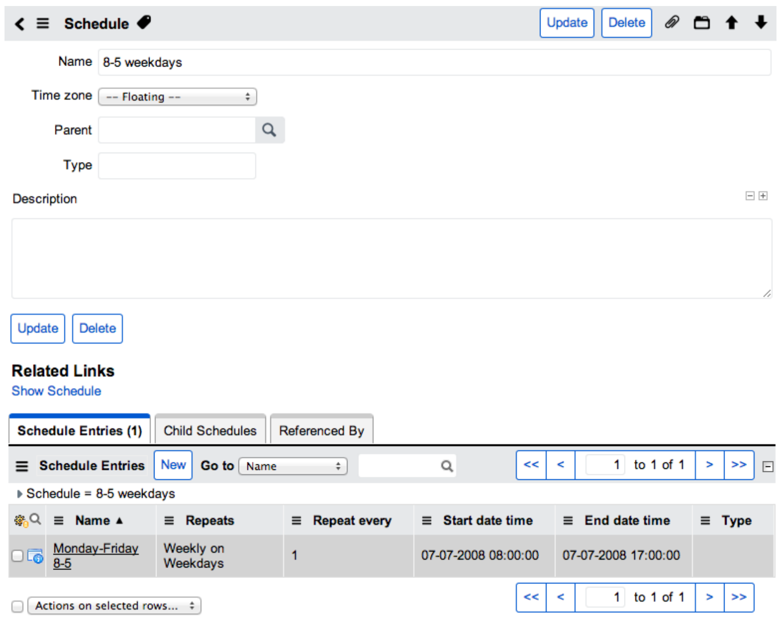
Schedules are configured with two types of records:

- *Schedule* records specify a time zone and a type of schedule and use one or more schedule entries. Schedule records are saved in the Schedule [cmn_schedule] table.
- *Schedule entry* records specify the time periods that are included or excluded from a schedule. Schedule entries are saved in the Schedule Entry [cmn_schedule_span] table.

Creating a Schedule

1. Navigate to **System Scheduler > Schedules > Schedules**.
2. Select a pre-existing schedule or click **New** to create a new one.
3. Complete the fields on the form (see table).

   Note: The Schedule form displays a warning message if there are no active entries defined for the current schedule. If your schedule is a child schedule that only contains exclusions, ignore the message because exclusions are non-active entries.
4. Right-click the header bar and click **Save**.
5. Configure one or more schedule entries.



Schedule

Update Delete

Name	8-5 weekdays
Time zone	-- Floating --
Parent	
Type	
Description	

Update Delete

Related Links

Show Schedule

Schedule Entries (1) | Child Schedules | Referenced By

Schedule Entries New Go to Name

1 to 1 of 1

Schedule = 8-5 weekdays

Name ▲	Repeats	Repeat every	Start date time	End date time	Type
Monday-Friday 8-5	Weekly on Weekdays	1	07-07-2008 08:00:00	07-07-2008 17:00:00	

Actions on selected rows...

1 to 1 of 1

The default Schedule form for the 8-5 weekdays schedule.

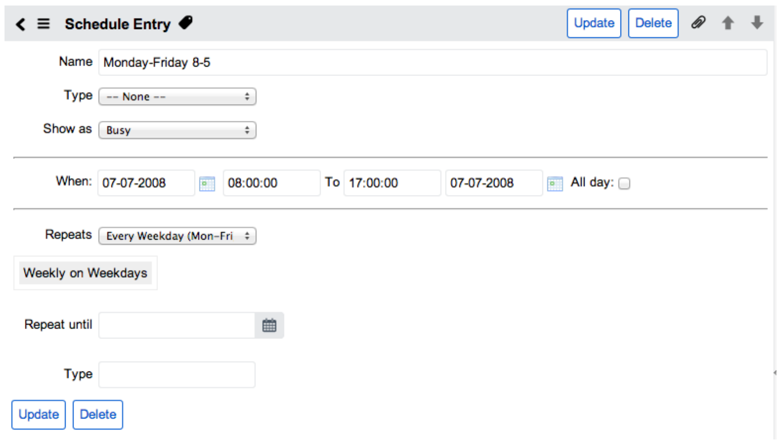
Field	Description
Name	Enter a unique name for the schedule.
Time Zone	Select the time zone for the schedule. If you select **Floating**, the time zone will be relative to whatever is accessing the item at any given time. For example, if a resource manager in Amsterdam sets a floating schedule for 8:00A.M.–5:00P.M., a user in San Jose sees the schedule as 8:00 A.M.–5:00 P.M. When a schedule is defined in a specific time zone, users in different time zones see the schedule with their own time zone applied.
Parent	Select a parent schedule to constrain the new schedule.
Type	Enter a label that describes the purpose of the schedule. You can also use one of these system terms to determine how to process certain schedules: • **excluded:** excludes time periods from SLA counts. • **maintenance:** specifies time periods where change management activities are allowed. A schedule containing maintenance schedule entries cannot also contain blackout schedule entries. • **blackout:** excludes time periods from change management schedules. A schedule containing blackout schedule entries cannot also contain maintenance schedule entries.
Description	[Optional] Describe the schedule.

Note: *If you create a schedule of type **maintenance** and save the record, a UI policy hides the **Type** field from the form. To view or change the value for the **Type** field, view the list of schedules rather than the schedule form and add the **Type** column if necessary. You can double click the cell for the value in the **Type** column and modify from the list view.*

Creating a Schedule Entry

1. Open a schedule record.
2. In the **Schedule Entries** related list, click **New**.
3. Complete the fields on the form (see table) to add a time period that is either included (for example, **Show as** is **Busy**) or excluded (**Type** is **Excluded**).
4. Click **Submit**.
5. Create as many schedule entries as necessary.
6. [Optional] In the **Child Schedules** related list, click **Edit** and select the child schedule to include in this schedule. For example, select the **U.S. Holidays** schedule to exclude common U.S. holidays from your work schedule.

Note: *Schedules only include schedule entries from a parent and its direct child schedules. Schedule entries from a child schedule of another child schedule are not included in a parent schedule. For example, if schedules B and C both have schedule A as their parent schedule, then the schedule entries for both schedules B and C are included in schedule A. However, if the parent schedule of schedule C is schedule B, the schedule entries for schedule C are excluded from schedule A.*



Name	Monday-Friday 8-5
Type	-- None --
Show as	Busy
When:	07-07-2008 08:00:00 To 17:00:00 07-07-2008 All day: ☐
Repeats	Every Weekday (Mon-Fri
Weekly on Weekdays	
Repeat until	
Type	

The default Schedule Entry form used by the 8-5 weekdays schedule.

Field	Description
Name	Enter a unique name for the schedule entry.
Type	Select the type of schedule entry this applies to.
Show As	Select an option to indicate how the schedule entry should be displayed in calendar applications and how it should interact with other schedule entries.
When	Enter the date and time to which the schedule entry applies. If the schedule entry applies to a full 24-hour day, select the **All day** check box.
Repeats	Select a repetition interval for the schedule entry, if any. If you select a repetition interval, ServiceNow displays other fields to further specify the repeat interval.
Repeat every	Select how often the schedule repeats daily, weekly, monthly, or yearly. This field is only visible when the **Repeats** field has a value of **Daily**, **Weekly**, **Monthly**, or **Yearly**.
Repeat on	Select the days of the week a weekly schedule repeats on. This field is only visible when the **Repeats** field has a value of **Weekly**.
Monthly type	Select how a monthly schedule repeats. This field is only visible when the **Repeats** field has a value of **Monthly**. Monthly repeat options include: • Repeat on a specific day of the month • Repeat on a specific day in a specific week of the month • Repeat on the last day of the month • Repeat on a specific week day in the last week of the month

Yearly type	Select how a yearly schedule repeats. This field is only visible when the **Repeats** field has a value of **Yearly**. Yearly repeat options include: • Repeat on a specific day of the year • Repeat on a floating day
Float week	Select which week of the month a floating yearly schedule repeats on. This field is only visible when the **Yearly type** field has a value of **Floating**.
Float day	Select which day of the week a floating yearly schedule repeats on. This field is only visible when the **Yearly type** field has a value of **Floating**.
Month	Select which month of the year a floating yearly schedule repeats on. This field is only visible when the **Yearly type** field has a value of **Floating**.
Repeat until	Select a repetition end date. If you leave this field blank, the schedule repeats indefinitely.
Type	[Optional] Enter a schedule entry description or use one of these system terms to determine how to process certain schedules. See Creating a Schedule for descriptions of the system terms. This field does not appear by default for new schedule entries, but you can configure the form to add it. If you enter **Excluded** in this field, the entire schedule entry is excluded from the schedule.

Repeating Monthly Schedules

For monthly schedules (**Repeat** is set to **Monthly**) that start on a particular day of the month (**Monthly type** is set to **Day of the month**), you can specify the following options:

- How ServiceNow computes the starting day each month. See Day of the Week.
- How ServiceNow handles monthly schedules that start on the fifth instance of a day. See Fifth Instance of a Day of the Week.

Day of the Week

ServiceNow offers two methods to compute what day of the week a monthly schedule repeats on:

- **Day:** This method computes the day of the week to repeat on by determining the order of the selected starting date within the month. For example, if the selected starting date appears on the first Monday in the month, the schedule repeats every first Monday of every month.
- **Week:** This legacy method computes the day of the month to repeat on by determining what week number the selected starting date appears in the month. For example, if the starting date is a Monday during the second week of the month, the schedule repeats the second Monday of every month.

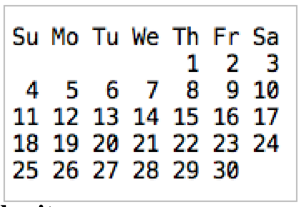
The system property `glide.schedules.repeat_nth` determines what method your instance uses to compute what day a repeating monthly schedule occurs on. By default, instances use the more accurate **Day** method.

Note: *Use the* ***Week*** *method to maintain backwards compatibility with customized schedule logic.*

The following example illustrates computing what day of the week a monthly schedule repeats on.

1. Navigate to **sys_properties.list**.
2. Open the `glide.schedules.repeat_nth` property.
3. Verify that the **Value** is set to **day**.
4. Navigate to **System Scheduler > Schedules > Schedules**, define a new schedule, and click **Submit**.
5. Open the new schedule and in the **Schedule Entries** related list, create a new entry with the following parameters:
    - **When:** *November 5, 2012 at 10:00 to November 5, 2012 at 11:00*

- **Repeats:** *Monthly*
- **Monthly type:** *Day of the Week*
- **Starting:** *November 5* (note that November 5 is the first Monday in the month, but it is in the second week)



Su	Mo	Tu	We	Th	Fr	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

6. Click **Submit**.
7. Open the **Schedule Entry**.

   Note that the form says **Every month on the first Mon.**

   The first few dates this schedule will run are:

   - November 5, 2012 (1st Monday of the month)
   - December 3, 2012 (1st Monday of the month)
   - January 7, 2012 (1st Monday of the month)
8. If the **Value** on the `glide.schedules.repeat_nth` property is set to **week** instead of **day** in step 3, the first few dates this schedule will run are:
   - November 5, 2012 (Schedule starts on Monday in the 2nd week of the month)
   - December 10, 2012 (2nd Monday in the month)
   - January 14, 2012 (2nd Monday in the month)

Fifth Instance of a Day of the Week

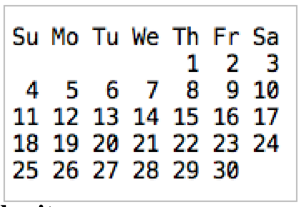
When selecting a date near the end of a month for a repeating monthly schedule, it is possible to select a date that computes to the fifth instance of that week day. ServiceNow offers three options for handling months that do not have a matching fifth instance of the selected day.

- **Last:** ServiceNow selects the last instance of the week day in the month.
- **Next:** ServiceNow selects the first instance of the week day in the next month.
- **Strict:** ServiceNow skips any month without a matching fifth instance and selects only months that have a matching fifth instance.

The system property `glide.schedules.fifth` controls how a schedule entry that selects the fifth occurrence of a week day behaves in months containing only four occurrences of that day. This property is only valid when the `glide.schedules.repeat_nth` property is set to **Day**.

The following example illustrates computing what day of the month a schedule repeats on when the schedule starts on the fifth instance of a week day in the month.

1. Navigate to **sys_properties.list**.
2. Open the `glide.schedules.fifth` property.
3. Verify that the **Value** is set to **last**.
4. Navigate to **System Scheduler > Schedules > Schedules**, define a new schedule, and click **Submit**.
5. Open the new schedule and in the **Schedule Entries** related list, create a new entry with the following parameters:
   - **When:** *November 29, 2012 at 10:00 to November 29, 2012 at 11:00*
   - **Repeats:** *Monthly*
   - **Monthly type:** *Day of the Week*
   - **Starting:** *November 29* (note that November 29 is the fifth Thursday in the month)



```
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

6. Click **Submit**.
7. Open the same schedule entry.

   Note that the form says "Every month on the fifth Thu."

   The schedule for the first three months is computed as:

   - November 29, 2012 (5th Thursday of the month)
   - December 27, 2012 (Last Thursday of the month)
   - January 31, 2013 (5th Thursday of the month)
8. If the **Value** on the `glide.schedules.fifth` property is set to **next** instead of **last** in step 3, the schedule for the first three months is computed as:
   - November 29, 2012 (5th Thursday of the month)
   - January 3, 2012 (1st Thursday of the next month since December 2012 does not have five Thursdays)
   - January 31, 2013 (5th Thursday of the month)
9. If the **Value** on the `glide.schedules.fifth` property is set to **strict** instead of **last** in step 3, the schedule for the first three months is computed as:
   - November 29, 2012 (5th Thursday of the month)
   - No meeting (December 2012 skipped because it does not have five Thursdays)
   - January 31, 2013 (5th Thursday of the month)

Parent and Child Schedules

Schedules can have one of two parent-child relationship with other schedules.

- **Parent field:** When a schedule record lists a value for the **Parent** field, schedule entries from the parent schedule apply to both the parent schedule and the child schedule. By default, there are no sample schedules that use the **Parent** field.
- **Child schedule:** When a schedule record has one or more child schedules in the **Child Schedules** related list, schedule entries from the child schedule apply to the containing schedule. By default, there are several sample schedules that use child schedules. For example, see the **8-5 weekdays excluding holidays** schedule that includes the **U.S. Holidays** schedule.

Parent and child schedules cannot contain conflicting schedule entry types. For example, a schedule containing maintenance schedule entries cannot also contain blackout schedule entries. Nor can a maintenance schedule have a child schedule containing blackout schedule entries.

Parent schedules are not valid if they are only exclusionary. They must have at least one entry that is not of type **Excluded**.

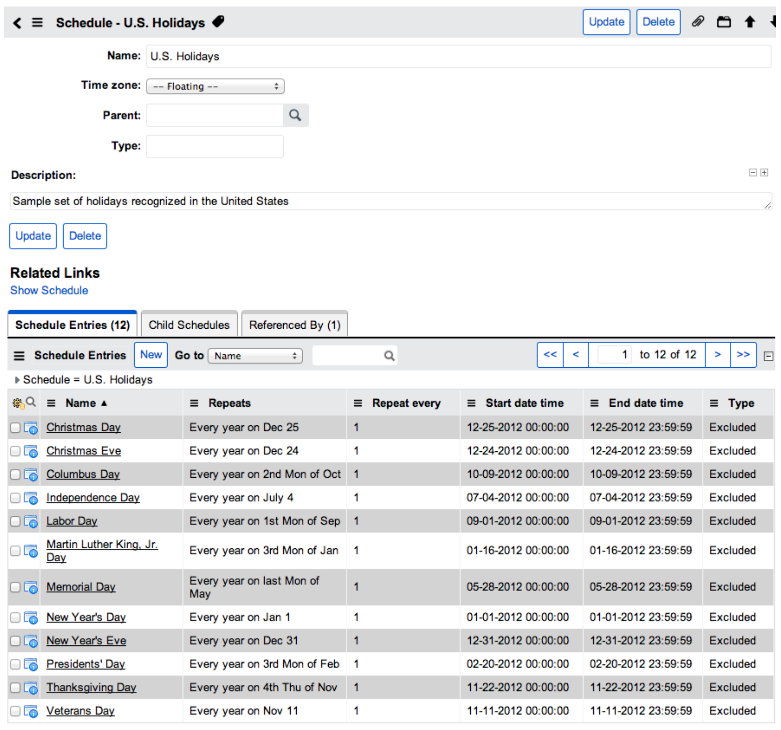
Note: *The* ***Show Schedule*** *related link only shows schedule entries from the current schedule record. The view does not include any schedule entries from related child schedules. For example, when showing the* ***8-5 weekdays excluding holidays*** *schedule, holidays do not show as excluded because the holiday schedule is a child schedule.*

Holidays

Each individual holiday can be defined as a schedule entry to create exceptions to existing schedules. For instance, if an SLA requires an incident be resolved within three business days excluding Christmas, create a schedule entry for Christmas to ensure that SLAs do not count Christmas when calculating elapsed time, even if it falls within the work week.

Because schedules can be included in other schedules through a parent-child relationship, it is also possible to create a holiday schedule and include it in other schedules to keep holidays consistent.

The following example shows a holiday schedule.



Name:	U.S. Holidays
Time zone:	-- Floating --
Parent:	
Type:	

Description:

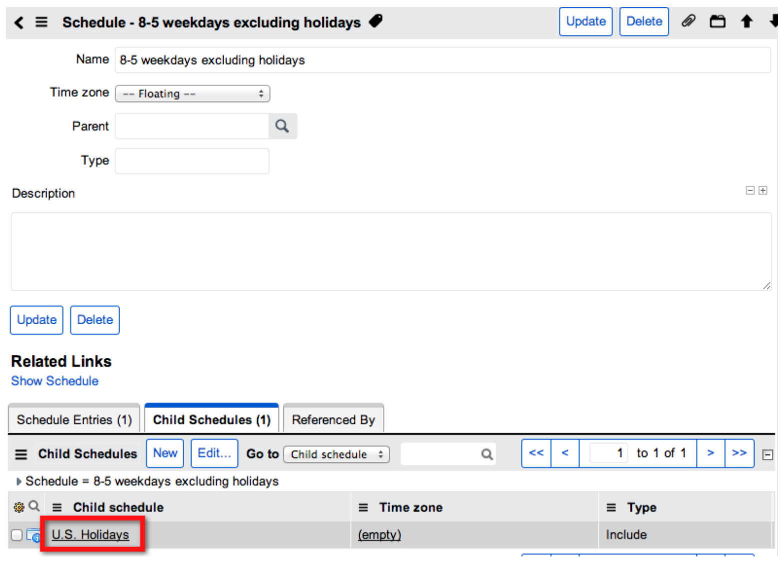
Sample set of holidays recognized in the United States

Schedule = U.S. Holidays

Name ▲	Repeats	Repeat every	Start date time	End date time	Type
Christmas Day	Every year on Dec 25	1	12-25-2012 00:00:00	12-25-2012 23:59:59	Excluded
Christmas Eve	Every year on Dec 24	1	12-24-2012 00:00:00	12-24-2012 23:59:59	Excluded
Columbus Day	Every year on 2nd Mon of Oct	1	10-09-2012 00:00:00	10-09-2012 23:59:59	Excluded
Independence Day	Every year on July 4	1	07-04-2012 00:00:00	07-04-2012 23:59:59	Excluded
Labor Day	Every year on 1st Mon of Sep	1	09-01-2012 00:00:00	09-01-2012 23:59:59	Excluded
Martin Luther King, Jr. Day	Every year on 3rd Mon of Jan	1	01-16-2012 00:00:00	01-16-2012 23:59:59	Excluded
Memorial Day	Every year on last Mon of May	1	05-28-2012 00:00:00	05-28-2012 23:59:59	Excluded
New Year's Day	Every year on Jan 1	1	01-01-2012 00:00:00	01-01-2012 23:59:59	Excluded
New Year's Eve	Every year on Dec 31	1	12-31-2012 00:00:00	12-31-2012 23:59:59	Excluded
Presidents' Day	Every year on 3rd Mon of Feb	1	02-20-2012 00:00:00	02-20-2012 23:59:59	Excluded
Thanksgiving Day	Every year on 4th Thu of Nov	1	11-22-2012 00:00:00	11-22-2012 23:59:59	Excluded
Veterans Day	Every year on Nov 11	1	11-11-2012 00:00:00	11-11-2012 23:59:59	Excluded

A schedule for U.S. holidays.

The following example shows a schedule that includes the holiday schedule shown above.


Schedule - 8-5 weekdays excluding holidays

Name	8-5 weekdays excluding holidays
Time zone	-- Floating --
Parent	
Type	
Description	

Related Links
Show Schedule

Schedule Entries (1) | Child Schedules (1) | Referenced By

Child Schedules — Go to Child schedule — 1 to 1 of 1

Schedule = 8-5 weekdays excluding holidays

Child schedule	Time zone	Type
U.S. Holidays	(empty)	Include

The U.S. holiday schedule used by another schedule.

Creating Holiday Schedules for Multiple Regions

You can use the following method to support multiple regions that all follow the same work schedule (for example, an 8-5 weekdays schedule) but have different holiday schedules.

1. Create a holiday schedule for each region. For example, U.S. Holidays, British Holidays, and Australian Holidays.
2. Add the work schedule as a child schedule to each region's holiday schedule.

This method requires making <number of schedules> + 1 total schedules. If you instead make the regional holiday schedule a child schedule of the work hours schedule, you will need to create a separate work hours schedule for each region. The total number of schedules in this case is <number of schedules> x 2 schedules.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/concept/c_UseSchedules.html

Displaying Time

Displaying Time

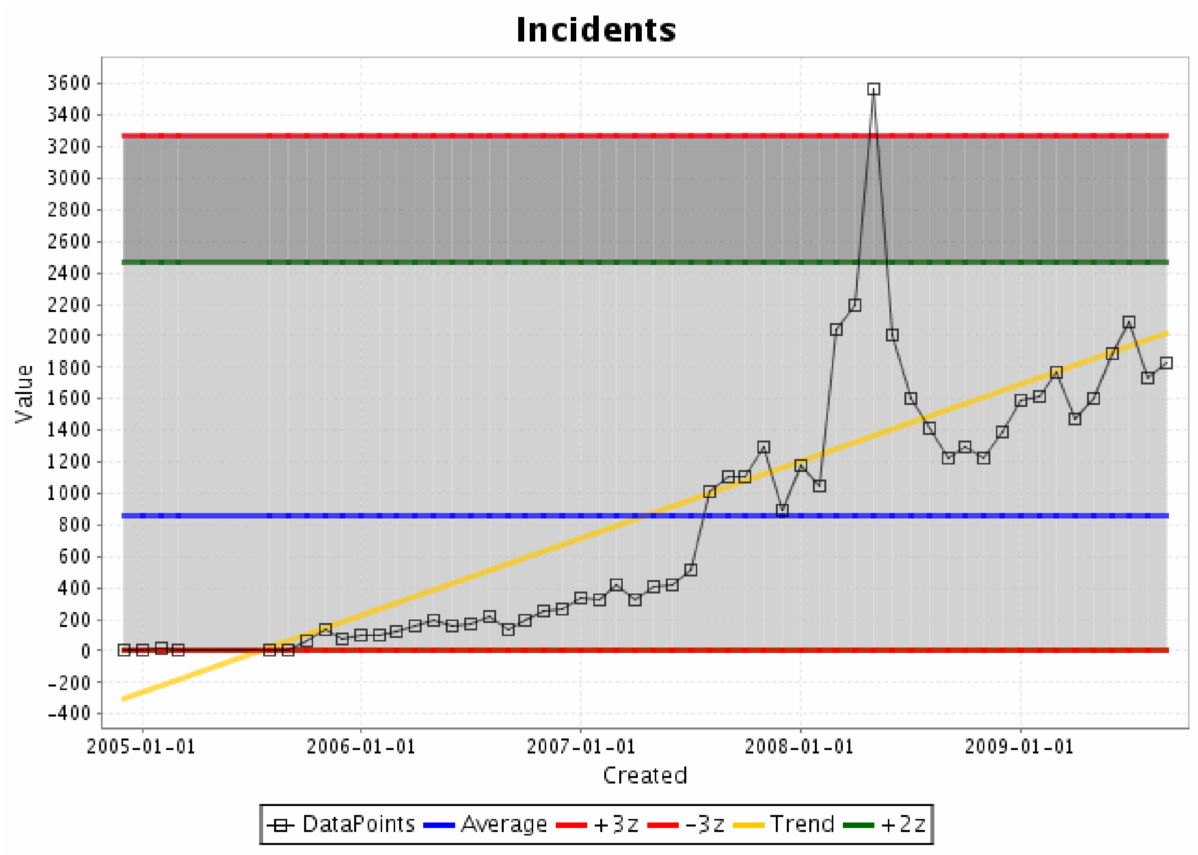
Note: *This article applies to Fuji and earlier releases. For more current information, see Time Display* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**'

Overview

There are three interfaces provide for displaying record information over time:

- Reports
- Timeline Pages
- Schedule Pages

Reports



Reporting allows users to generate charts from data within the platform in a variety of formats.

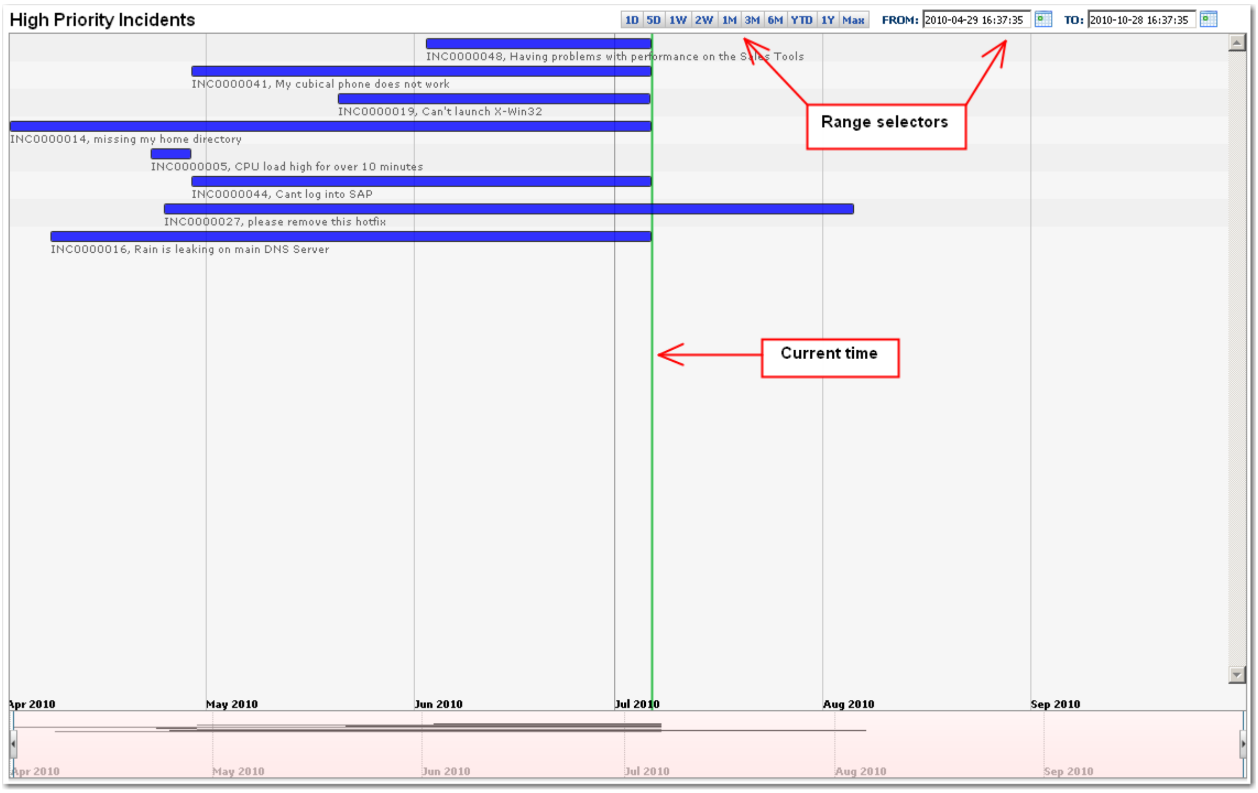
Any report type can report in time information, but the following report types are particularly suited to reporting on time information:

- Calendars

- Control Charts
- Trend Charts
- Trendbox Charts

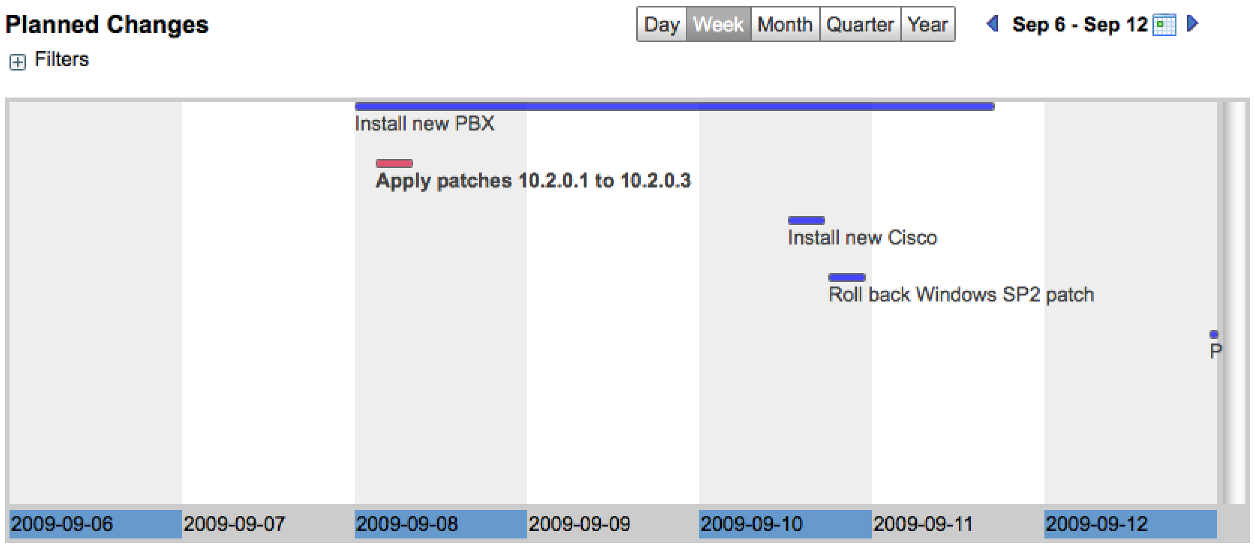
For more information on the types of reports available, see Report Types.

Timeline Pages


High Priority Incidents
1D 5D 1W 2W 1M 3M 6M YTD 1Y Max
FROM: 2010-04-29 16:37:35
TO: 2010-10-28 16:37:35
INC0000048, Having problems with performance on the Sales Tools
INC0000041, My cubical phone does not work
INC0000019, Can't launch X-Win32
INC0000014, missing my home directory
INC0000005, CPU load high for over 10 minutes
INC0000044, Cant log into SAP
INC0000027, please remove this hotfix
INC0000016, Rain is leaking on main DNS Server
Range selectors
Current time
Apr 2010
May 2010
Jun 2010
Jul 2010
Aug 2010
Sep 2010

Timeline pages allow for easy definition of linear timelines from records with time information. For more information, see Timeline Pages.

Schedule Pages



Schedule Pages can defines Timelines or Gantt Charts for visualizing tasks over time. These interfaces provide more tools for modifying the tasks using drag and drop tools. For more information, see Schedule Pages.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/reference/r_TimeDisplay.html

Schedule Pages

Note: *This article applies to Fuji and earlier releases. For more current information, see Schedule Pages* [1] *at* http://docs. servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**

Overview

A schedule page is a record that contains a collection of scripts that allow for custom generation of a calendar or timeline display. For a discussion of how calendars are created using Schedule Pages, see **Creating Calendars with Schedule Pages**. For information about custom generated timelines using schedule pages, see **Timelines**. See **Application Programming Interface (API)** for links to classes and methods in the API.

Schedule Pages Form

To access Schedule Pages, navigate to **System Scheduler > Schedules > Schedule Pages**. The form provides the following fields, depending upon the **View Type** selected:

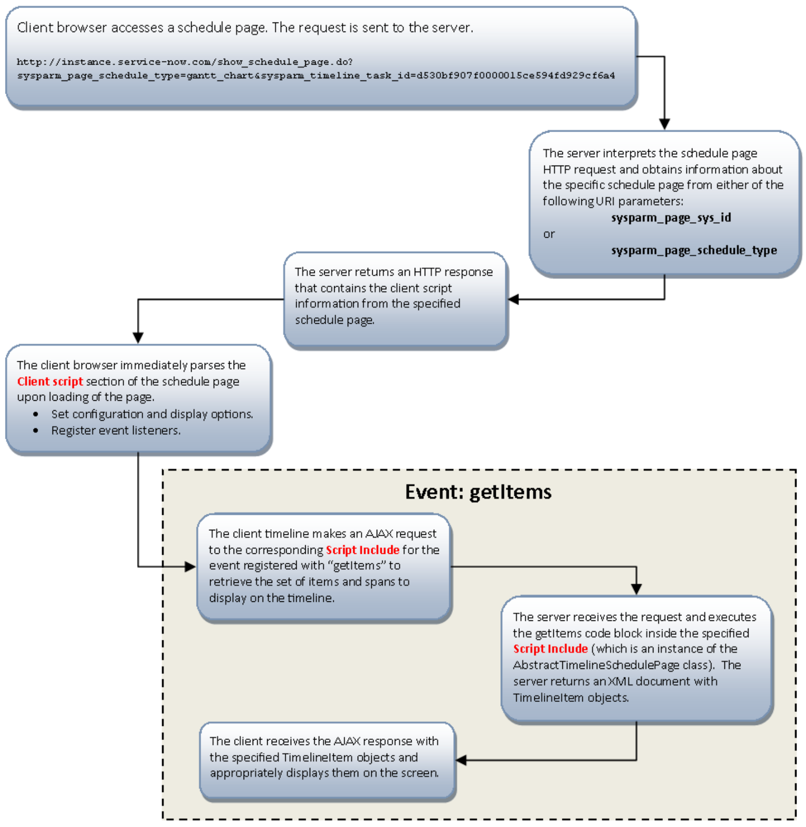
Field	Field Type	Description
Name	String	General name that is used to identity the current schedule page.
Schedule type	String	The schedule type is a string that is used to uniquely identity the schedule page via the "sysparm_page_schedule_type" URI parameter. For example, a schedule page could be accessed as follows: /show_schedule_page.do?**sysparm_page_schedule_type**=gantt_chart&sysparm_timeline_task_id=d530bf907f0000015ce594fd929cf6a4 Alternatively, the schedule page can also be accessed by setting the "sysparm_page_sys_id" URI parameter to that of the unique 32 character hexadecimal system identifier of the schedule page.
View Type	Choice	Each view type displays different field combinations. There are two options available : • **Calendars** • **Timelines**
Description	String	General description that provides additional information about the current schedule page. This field is not necessary.
Init function name	String	**Note:** *This functionality is only used by* ***Calendar*** *type schedule pages.* The init function name specifies the name of the JavaScript function to call inside the **Client script** function for calendar type schedule pages.
HTML	String	**Note:** *This functionality is only used by* ***Calendar*** *type schedule pages.* The HTML field is a scriptable section that is parsed by Jelly and injected into the display page prior to the rest of the calendar. It can be used to pass in variables from the server and define extra fields are necessary.

Client script	String	The client script is a scriptable section that allows for configuring options of the schedule page display. The API is different depending on the schedule page view type and is discussed below.
Server AJAX processor	String	**Note:** *This functionality is only used by* ***Calendar*** *type schedule pages.* The Server AJAX processor is specific to calendar type schedule pages that is used to return a set of schedule items and spans to be displayed.

Timelines

A Timeline Schedule Page is a specific record that contains configuration information for displaying time based points and spans in a "timeline" like fashion. The timeline schedule page references a script include that extends from `AbstractTimelineSchedulePage` to perform dynamic modification to the timeline based on different events and conditions. Both the schedule page and the script include for timeline generation allow for extreme customization and their corresponding application programming interface (API) is documented below.

The following diagram shows the series of events that occur when a timeline schedule page is accessed. Once the timeline has been loaded, all subsequent events, such as events resulting from timeline interaction (e.g. moving a timeline span), follow the same logic flow shown in the gray event box.


Client browser accesses a schedule page. The request is sent to the server.
http://instance.service-now.com/show_schedule_page.do?sysparm_page_schedule_type=gantt_chart&sysparm_timeline_task_id=d530bf907f0000015ce594fd929cf6a4
The server interprets the schedule page HTTP request and obtains information about the specific schedule page from either of the following URI parameters: sysparm_page_sys_id or sysparm_page_schedule_type
The server returns an HTTP response that contains the client script information from the specified schedule page.
The client browser immediately parses the Client script section of the schedule page upon loading of the page.
• Set configuration and display options.
• Register event listeners.
Event: getItems
The client timeline makes an AJAX request to the corresponding Script Include for the event registered with "getItems" to retrieve the set of items and spans to display on the timeline.
The server receives the request and executes the getItems code block inside the specified Script Include (which is an instance of the AbstractTimelineSchedulePage class). The server returns an XML document with TimelineItem objects.
The client receives the AJAX response with the specified TimelineItem objects and appropriately displays them on the screen.

Applications that Use Schedule Pages to Generate Timelines

- **Project Management**
- **Maintenance Schedules**
- **Group On-Call Rotation**
- **Field Service Management**

Application Programming Interface (API)

Timeline schedule pages use an internal class that allows for direct customization of how a timeline is displayed. Creation of new timeline schedule pages requires thorough understanding of how the page/event flow [link to above] works as well as capability to write both client and server side JavaScript. The API described below details the available methods and scripting requirements for both the client side and server side.

Note: ***Click here*** *to view a complete example of a new timeline schedule page with its corresponding script include.*

The timeline Application Programming Interface (API) contains the following classes:

- **Client Script / Class GlideTimeline**
- **Script Include / Class AbstractTimelineSchedulePage**
- **Class GlideTimelineItem**
- **Class TimelineSpan**

See the Incident Summary Timeline **example** for information about how a project support manager might use these APIs to create a timeline in which new incidents are grouped together by priority and closed incidents by duration.

Client Script / Class GlideTimeline

The GlideTimeline class provides the core implementation for configuring and displaying a Glide Windowing Toolkit Timeline. For security the GlideTimeline has already been instantiated as a single instance variable named: **glideTimeline**. All configurations should be made in the client script section of the corresponding schedule page referencing this instance variable.

Method Summary

Return Value	Details
void	**setReadOnly** `(Boolean b)` Enables or disables all timeline event interaction. If enabled, event interaction is determined from the corresponding attributes specified by each Timeline Item. The default value for the readOnly property is `false`.
void	**showLeftPane** `(Boolean b)` Specifies whether or not to show the left hand pane in the timeline. The default value for the showLeftPane property is `true`.
void	**showSummaryPane** `(Boolean b)` Specifies whether or not to show the summary pane at the bottom of the timeline. The default value for the showSummaryPane property is `true`.
void	**showLeftPaneAsTree** `(Boolean b)` Specifies whether or not to show items that have a parent attribute as nested indented nodes with expand and collapse capability. The default value for the showLeftPaneAsTree property is `false`.

void	**showTimelineText** `(Boolean b)` Specifies whether or not to show the timeline text underneath each Timeline Span in the primary timeline pane. The default value for the showTimelineText property is `false`.
void	**showDependencyLines** `(Boolean b)` Specifies whether or not to show dependency lines between Timeline Spans. This is only applicable if the set of Timeline Items returned from the server includes dependency relationships. The default value for the showDependencyLines property is `false`.
void	**groupByParent** `(Boolean b)` Specifies whether or not to group items by their parent. If `true`, this will nest all child items inside their parent. This affects the ordering of display and children will always be listed immediately after their parent. The default value for the groupByParent property is `false`.
void	**sortByLeftLabelText** `(Boolean b)` Specifies whether or not to sort the list of Timeline Items returned in alphabetical order according to the text that was specified to show in the Left Pane. Note that this function will still sort regardless if the left pane is disabled. Additionally, if groupByParent() is set `true`, items will be sorted appropriately after grouping has occurred. The default value for the sortByLeftLabelText property is `false`.
void	**sortByTimelineLabelText** `(Boolean b)` Specifies whether or not to sort the list of Timeline Items returned in alphabetical order according to the text that was specified to show in the Timeline Pane. Note that this will still sort will still occur regardless if the timeline text has been set false via the showTimelineText() method. Additionally, if groupByParent() is set `true`, items will be sorted appropriately after grouping has occurred. The default value for the sortByTimelineLabelText property is `false`.
void	**sortByStartDate** `(Boolean b)` Specifies whether or not to sort the list of Timeline Items returned by the earliest start date of an item's Timeline Span objects. Note that if groupByParent() is set `true`, items will be sorted appropriately after grouping has occurred. The default value for the sortByStartDate property is `false`.
void	**setDefaultPointIconClass** `(String strName)` Specifies the default icon class to use for Timeline Spans with zero duration if no icon class was explicitly specified in the properties of the Timeline Span returned from the server. The default value for the setDefaultPointIconClass property is `milestone`.
void	**showLeftPaneInputBox** `(Boolean b, String strDefaultValue)` Specifies whether or not to show the text input box at the bottom of the left pane with a default value as specified by `strDefaultValue`. Note that if the left pane is diabled via showLeftPane() the input box will not be visible. The default value for the showLeftPaneInputBox property is `false`.
void	**setInitialViewRange** `(mixed objStartDate, mixed objEndDate)` Specifies the initial viewable range for the timeline. The format of the start and end date must be either in the default timestamp format: `yyyy-MM-dd HH:mm:ss` or specified as a number representing the time in milliseconds since the standard epoch of 1/1/1970. The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the initialViewRange property is specified, it will override the default range.
void	**setExtraAjaxParam** `(String strName, String strValue)` Allows setting of additional parameters in the client script to be made available to the corresponding Script Include events by using the getParameter() method. URI parameters that are prefixed with "sysparm_timeline_" will automatically be included in all server side AJAX calls.
void	**registerEvent** `(String strServerEvent, String strScriptIncludeName)` Registers the specified Timeline server event. The `strServerEvent` must be one of the allowed events for registration to work correctly. When the event occurs, the GlideTimeline will send a request to the server and process the event as handled inside the `strScriptIncludeName` class.
void	**snapVertScrollingIntoRows** `(Boolean boolSnapVertScrollingIntoRows)` Specifies whether or not the vertical movement of timeline span objects (if appropriately registered to perform this event) should snap adjust into the closest row. By default this value is enabled.

void	**showGridLines** `(Boolean boolShowGridLines)` Specifies whether or not to show grid lines for each row of data on the timeline. By default, grid lines are enabled.
void	**setAutoRefresh** `(Number intSeconds)` Specifies the number of seconds to wait before performing an auto refresh of the data on the timeline. Setting the number of seconds to 0 will turn auto refresh off. By default, auto refresh is not enabled. Note that if `intSeconds` is greater than 0 and less than the minimum allowed time in seconds (10), it will be set to 10 seconds.

Back to Application Programming Interface (API)

setReadOnly

public void setReadOnly `(Boolean b)`

Enables or disables all timeline event interaction. If enabled, event interaction is determined from the corresponding attributes specified by each Timeline Item. The default value for the readOnly property is `false`.

Parameters:

`b` - Boolean variable that marks the entire timeline as read-only (non-interactive) if set to `true`.

Example:

```
glideTimeline.setReadOnly(true);
```

showLeftPane

public void showLeftPane `(Boolean b)`

Specifies whether or not to show the left hand pane in the timeline. The default value for the leftPane property is `true`.

Parameters:

`b` - Boolean variable that will show the left pane if set `true`; otherwise, the left pane will not be displayed.

Example:

```
glideTimeline.showLeftPane(false);
```

showSummaryPane

public void showSummaryPane `(Boolean b)`

Specifies whether or not to show the summary pane at the bottom of the timeline. The default value for the showSummaryPane property is `true`.

Parameters:

`b` - Boolean variable that will show the summary pane if set `true`; otherwise, if `false` the summary pane will not be displayed.

Example:

```
glideTimeline.showSummaryPane(false);
```

showLeftPaneAsTree

public void showLeftPaneAsTree(`Boolean b`)

Specifies whether or not to show items that have a parent attribute as nested indented nodes with expand and collapse capability. The default value for the showLeftPaneAsTree property is `false`.

Parameters:

`b` - Boolean variable that will display child item nodes indented with expand/collapse capability if set `true`; otherwise, all left pane items will be displayed at a single indent level.

Example:

```
glideTimeline.showLeftPaneAsTree(true);
```

showTimelineText

public void showTimelineText(`Boolean b`)

Specifies whether or not to show the timeline text underneath each Timeline Span in the primary timeline pane. The default value for the showTimelineText property is `false`.

Parameters:

`b` - Boolean variable that will display descriptive text underneath each Timeline Span if set `true`; otherwise, no text will be displayed underneath each Timeline Span.

Example:

```
glideTimeline.showTimelineText(true);
```

showDependencyLines

public void showDependencyLines(`Boolean b`)

Specifies whether or not to show dependency lines between Timeline Spans. This is only applicable if the set of Timeline Items returned from the server includes dependency relationships. The default value for the showDependencyLines property is `false`.

Parameters:

`b` - Boolean variable that will display dependency lines on the timeline if set `true`; otherwise, will not.

Example:

```
glideTimeline.showDependencyLines(true);
```

groupByParent

public void groupByParent(`Boolean b`)

Specifies whether or not to group items by their parent. If `true`, this will nest all child items inside their parent. This affects the ordering of display and children will always be listed immediately after their parent. The default value for the groupByParent property is `false`.

Parameters:

`b` - Boolean variable that will group Timeline Items by their parent if set `true`

Example:

```
glideTimeline.groupByParent(true);
```

sortByLeftLabelText

public void sortByLeftLabelText`(Boolean b)`

Specifies whether or not to group items by their parent. If `true`, this will nest all child items inside their parent. This affects the ordering of display and children will always be listed immediately after their parent. The default value for the groupByParent property is `false`.

Parameters:

`b` - Boolean variable that will sort Timeline Items alphabetically by the text specified in an item's left label if set `true`

Example:

```
glideTimeline.sortByLeftLabelText(true);
```

sortByTimelineLabelText

public void sortByTimelineLabelText`(Boolean b)`

Specifies whether or not to sort the list of Timeline Items returned in alphabetical order according to the text that was specified to show in the Timeline Pane. Note that this will still sort will still occur regardless if the timeline text has been set false via the showTimelineText() method. Additionally, if groupByParent() is set `true`, items will be sorted appropriately after grouping has occurred. The default value for the sortByTimelineLabelText property is `false`.

Parameters:

`b` - Boolean variable that will sort Timeline Items alphabetically by the text specified in an item's timeline span text if set `true`

Example:

```
glideTimeline.sortByTimelineLabelText(true);
```

sortByStartDate

public void sortByStartDate`(Boolean b)`

Specifies whether or not to sort the list of Timeline Items returned by the earliest start date of an item's Timeline Span objects. Note that if groupByParent() is set `true`, items will be sorted appropriately after grouping has occurred. The default value for the sortByStartDate property is `false`.

Parameters:

`b` - Boolean variable that will sort Timeline Items chronologically to their earliest start date if set `true`

Example:

```
glideTimeline.sortByStartDate(true);
```

setDefaultPointIconClass

public void sortByStartDate (`String strName`)

Specifies the default icon class to use for Timeline Spans with zero duration if no icon class was explicitly specified in the properties of the Timeline Span returned from the server. The default value for the setDefaultPointIconClass property is `milestone`.

Parameters:

`strName` - String that specifies one of the following values:

- `milestone` ◆
- `blue_square` 🟦
- `sepia_square` 🟫
- `green_square` 🟩
- `red_square` 🟥
- `black_square` ⬛
- `blue_circle` 🔵
- `sepia_circle` 🟤
- `green_circle` 🟢
- `red_circle` 🔴
- `black_circle` ⚫

Example:

```
glideTimeline.setDefaultPointIconClass('blue_circle');
```

showLeftPaneInputBox

public void showLeftPaneInputBox (`Boolean b, String strDefaultValue`)

Specifies whether or not to show the text input box at the bottom of the left pane with a default value as specified by `strDefaultValue`. Note that if the left pane is diabled via showLeftPane() the input box will not be visible. The default value for the showLeftPaneInputBox property is `false`.

Parameters:

`b` - Boolean variable that will show the left pane input box if set `true`.

`strDefaultValue` - String that specifies the default value to display in the input box.

Example:

```
glideTimeline.showLeftPaneInputBox(true, 'Add a new task ...');
```

setInitialViewRange

public void setInitialViewRange (`mixed objStartDate, mixed objEndDate`)

Specifies the initial viewable range for the timeline. The format of the start and end date must be in the default timestamp format: `yyyy-MM-dd HH:mm:ss`. The default range is the range that specifies the earliest Timeline Span point to the end of the latest Timeline Span. If the initialViewRange property is specified, it will override the default range.

Parameters:

`objStartDate` - Either a string that specifies the start time of the view range in format: `yyyy-MM-dd HH:mm:ss` or a number representing the start time in milliseconds.

`objEndDate` - Either a string that specifies the end time of the view range in format: `yyyy-MM-dd HH:mm:ss` or a number representing the end time in milliseconds.

Example:

```
// Sets the initial range to begin on June 20th, 2010 at 8:00 AM and
end on June 28th, 2010 at 2:00 PM UTC time.
glideTimeline.setInitialViewRange("2010-06-20 08:00:00",
1277647200000);
```

setExtraAjaxParam

public void setExtraAjaxParam `(String strStartDate, String strEndDate)`

Allows setting of additional parameters in the client script to be made available to the corresponding Script Include events by using the getParameter() method. URI parameters that are prefixed with "sysparm_timeline_" will automatically be included in all server side AJAX calls..

Parameters:

`strName` - String that specifies the URI parameter name.

`strValue` - String that specifies the value of `strName`.

Example:

```
glideTimeline.setExtraAjaxParam("sysparm_timeline_limit", "5");
```

registerEvent

public void registerEvent `(String strServerEvent, String strScriptIncludeName)`

Registers the specified Timeline server event. The `strServerEvent` must be one of the allowed events for registration to work correctly. When the event occurs, the GlideTimeline will send a request to the server and process the event as handled inside the `strScriptIncludeName` class.

Parameters:

`strServerEvent` - String that specifies one of the following **case-sensitive** events:

- `getItems`
- `elementMoveX`
- `elementMoveY`
- `elementMoveXY`
- `elementSuccessor`
- `elementTimeAdjustStart`
- `elementTimeAdjustEnd`
- `inputBox`
- `itemMove`

`strScriptIncludeName` - String that specifies the name of the class to receive the `strServerEvent`. This class should must be defined in a script include that extends `AbstractTimelineSchedulePage`.

Example:

```
glideTimeline.registerEvent("getItems", "TimelineGanttSchedulePage");
```

snapVertScrollingIntoRows

public void snapVertScrollingIntoRows `(Boolean b)`

Specifies whether or not the vertical movement of timeline span objects (if appropriately registered to perform this event) should snap adjust into the closest row. By default this value is enabled.

Parameters:

`b` - Boolean variable that will snap vertical movement into rows if set `true`; otherwise, items will move exactly with respect to the mouse.

Example:

```
glideTimeline.snapVertScrollingIntoRows(false);
```

showGridLines

public void showGridLines `(Boolean boolShowGridLines)`

Specifies whether or not to show grid lines for each row of data on the timeline. By default, grid lines are enabled.

Parameters:

`boolShowGridLines` - Boolean variable that will show grid lines if set `true`; otherwise, grid lines will not be displayed.

Example:

```
glideTimeline.showGridLines(false); // Disables grid lines.
```

setAutoRefresh

public void setAutoRefresh `(Number intSeconds)`

Specifies the number of seconds to wait before performing an auto refresh of the data on the timeline. Setting the number of seconds to 0 will turn auto refresh off. By default, auto refresh is disabled. Note that if `intSeconds` is greater than 0 and less than the minimum allowed time in seconds (10), it will be set to 10 seconds.

Parameters:

`intSeconds` - Integer that specifies time in seconds between auto-refreshing.

Example:

```
glideTimeline.setAutoRefresh(15); // Sets the interval for
auto-refreshing to 15 seconds.
```

Script Include / Class AbstractTimelineSchedulePage

The processing of all data displayed within a timeline is performed first by utilizing the corresponding function of the specified script include. Like other script includes, the language syntax is JavaScript and follows the default security constraints of this type of resource. However, in order to deal with the complexity of the different types of display options a helper class was created. The `AbstractTimelineSchedulePage` is an abstract class that must be extended and paired with it's corresponding schedule page to display a timeline. At a minimum, extending classes should override the `getItems()` method as this is the primary event handler for returning items to be displayed on the client.

The AbstractTimelineSchedulePage returns data to the client that is processed in two phases. The first phase processes the data making actual updates to the timeline. Immediately after (if configured) the second phase will display a success message box, error message box, or dialog message prompt. The available options in phase one for manipulating data include:

1. **Do Not Update Any Items** - This is the default behavior. Do not perform any of the remaining steps in phase one.
2. **Update With Specific Items** - This is done using: `add()`.
3. **Render The Timeline Using the getItems() Function** - This is done using: `setDoReRenderTimeline(true)`.

Note: *If both TimelineItems are returned and* `setDoReRenderTimeline` *is set to true, the system will ignore the* `setDoReRenderTimeline` *property and explicitly show only the TimelineItems that were added via the* `add()` *function.*

The available options in phase two, which control optional message boxes after phase one is complete include:

1. **Do Not Display Any Message Boxes** - This is the default behavior.
2. **Display a Success Dialog Box** - This is done using: `setStatusSuccess()`.
3. **Display an Error Dialog Box** - This is done using: `setStatusError()`.
4. **Display a Dialog Confirm Box** - This is done using: `setStatusPrompt()`.

For each dialog function, see the corresponding API below for proper implementation.

Note: *A script include class that extends* `AbstractTimelineSchedulePage` *will automatically receive all Uri parameters from the original Url whose prefix begins with "**sysparm_timeline_**". To access the values of these, use:* `this.getParameter("sysparm_timeline_`**`VARIABLE`**`");` *inside your extended class. This is useful if you need to display a schedule page from a dynamic element, such as from a context menu from a list. By passing in dynamic data via the Url the schedule page will auto-include these parameters inside the Ajax calls and therefore will be accessible inside the AbstractTimelineSchedulePage script includes.*

Method Summary

The following methods exist to facilitate data manipulation when extending the event handling methods in the following section.

Return Value	Details
void	**add**`(TimelineItem objItem)` Adds a TimelineItem object that will be returned to the client and appropriately displayed on the timeline.
void	**addSeparator**`()` Adds a horizontal frame separator into the list of timeline items. All future items added via `add()` will be added into the subsequent timeline frame.
void	**setPageTitle**`(String strTitle)` Specifies the text to display as the title of the timeline. The page title can be set (and updated) from any interactive event; however, is recommended to be set during the `getItems()` event.
void	**setDoReRenderTimeline**`(Boolean b)` Specifies whether or not to re-render all of the timeline items using the `getItems()` function.
void	**setStatusError**`(String strTitle, String strMessage)` Denotes the current event request with an error status that will display a dialog box with the specified title and message immediately in phase two of the GlideTimeline event processing.
void	**setStatusSuccess**`(String strTitle, String strMessage)` Denotes the current event request with a success status that will display a dialog box with the specified title and message immediately in phase two of the GlideTimeline event processing.

void	**setStatusPrompt** `(String strTitle, String strMessage, String strOkFunction, String strCancelFunction, String strCloseFunction)` Denotes the current event request with a prompt status that will display a confirmation dialog box with the specified title and message immediately in phase two of the GlideTimeline event processing. The confirmation box displays an "OK" and "Cancel" button that each generate new events that will call the specified functions as denoted in the parameter arguments. **Note that the custom defined functions for the OK, Cancel, and Close will receive the same parameter arguments as those for the current event.**

Abstract Event Handler Methods

The following methods correspond with the specified event thrown from the schedule page.

Return Value	Details
void	**getItems** `()` Event handler for returning schedule items to display on the timeline.
void	**elementMoveX** `(String spanSysId, String newStartDateTimeMs)` Event handler for when a timeline span moves horizontally.
void	**elementMoveY** `(String spanSysId, String itemSysId, String newItemSysId)` Event handler for when a timeline span moves vertically.
void	**elementMoveXY** `(String spanSysId, String itemSysId, String newItemSysId, String newStartDateTimeMs)` Event handler for when a timeline span moves both horizontally and vertically.
void	**elementSuccessor** `(sstring spanSysId, String newSuccSpanId)` Event handler for when a timeline relationship has been created between two spans.
void	**elementTimeAdjustStart** `(String spanSysId, String newStartDateTimeMs)` Event handler for when a timeline span's start date was modified.
void	**elementTimeAdjustEnd** `(String spanSysId, String newStartDateTimeMs)` Event handler for when a timeline span's end date was modified.
void	**inputBox** `(String strInputText)` Event handler for when a string was typed into the left pane input box.
void	**itemMove** `(String itemSysId, String newItemSysId)` Event handler for when a timeline row item was moved and dragged into another row item.

Back to Application Programming Interface (API)

add

public void add `(GlideTimelineItem objItem)`

Adds a GlideTimelineItem object that will be returned to the client and appropriately displayed on the timeline.

Parameters:

`objItem` - The GlideTimelineItem object to add to the timeline.

addSeparator

public void addSeparator `()`

Adds a horizontal frame separator into the list of timeline items. All future items added via `add()` will be added into the subsequent timeline frame.

Example:

```
// Inside of a script include that extends AbstractTimelineSchedulePage
this.addSeparator();
```

setPageTitle

public void setPageTitle `(String strTitle)`

Specifies the text to display as the title of the timeline. The page title can be set (and updated) from any interactive event; however, is recommended to be set during the `getItems()` event.

Parameters:

`strTitle` - String that specifies the text to be displayed on the title of the timeline.

setDoReRenderTimeline

public void setDoReRenderTimeline `(Boolean b)`

Specifies whether or not to re-render all of the timeline items using the `getItems()` function.

Parameters:

`b` - Boolean variable that will re-render the timeline by making a new event call to the server's `getItems()` handler if set `true`.

setStatusError

public void setStatusError `(String strTitle, String strMessage)`

Denotes the current event request with an error status that will display a dialog box with the specified title and message immediately in phase two of the GlideTimeline event processing.

Parameters:

`strTitle` - String that specifies the text to be displayed in the dialog box title.

`strMessage` - String that specifies the text to be displayed within the dialog box. The text can contain HTML formatting.

setStatusSuccess

public void setStatusSuccess `(String strTitle, String strMessage)`

Denotes the current event request with a success status that will display a dialog box with the specified title and message immediately in phase two of the GlideTimeline event processing.

Parameters:

`strTitle` - String that specifies the text to be displayed in the dialog box title.

`strMessage` - String that specifies the text to be displayed within the dialog box. The text can contain HTML formatting.

setStatusPrompt

public void setStatusPrompt(`String strTitle, String strMessage, String strOkFunction, String strCancelFunction, String strCloseFunction`)

Denotes the current event request with a prompt status that will display a confirmation dialog box with the specified title and message immediately in phase two of the GlideTimeline event processing. The confirmation box displays an "OK" and "Cancel" button that each generate new events that will call the specified functions as denoted in the parameter arguments. **Note that the custom defined functions for the OK, Cancel, and Close will receive the same parameter arguments as those for the current event.**

Parameters:

`strTitle` - String that specifies the text to be displayed in the dialog box title.

`strMessage` - String that specifies the text to be displayed within the dialog box. The text can contain HTML formatting.

`strOkFunction` - String that specifies the name of the function to call in the current script include class if the "OK" button is clicked.

`strCancelFunction` - String that specifies the name of the function to call in the current script include class if the "Cancel" button is clicked.

`strCloseFunction` - String that specifies the name of the function to call in the current script include class if the "Close" button is clicked.

Example:

```
var MyTimelineScriptIncludeClass = Class.create();
MyTimelineScriptIncludeClass.prototype =
Object.extendsObject(AbstractTimelineSchedulePage, {

  getItems: function() {
    //...
  },

  elementTimeAdjustEnd: function(spanSysId, newEndDateTimeMs) {
    // Display a status prompt dialog box
    this.setStatusPrompt('Confirm Action', 'Are you sure you want to do
 that?',
                        'this._myOkHandlerFunction',
                        'this._myCancelHandlerFunction',
                        'this._myCloseHandlerFunction');
  },

  _myOkHandlerFunction: function(spanSysId, newEndDateTimeMs) { // ...
},

  _myCancelHandlerFunction: function(spanSysId, newEndDateTimeMs) { //
... },

  _myCloseHandlerFunction: function(spanSysId, newEndDateTimeMs) { //
... }
};
```

getItems

public void getItems()

Event handler for returning schedule items to display on the timeline.

elementMoveX

public void elementMoveX `(String spanSysId, String newStartDateTimeMs)`

Event handler for when a timeline span moves horizontally.

Parameters:

`spanSysId` - String that specifies the sys ID of the current span that is being adjusted.

`newStartDateTimeMs` - String that specifies the new start time of the span in milliseconds. Make sure to parse the string using `parseInt()` prior to performing any numerical comparisons.

elementMoveY

public void elementMoveY `(String spanSysId, String itemSysId, String newItemSysId)`

Event handler for when a timeline span moves vertically.

Parameters:

`spanSysId` - String that specifies the sys ID of the current span that is being adjusted.

`itemSysId` - String that specifies the sys ID of the timeline item associated with the current span.

`newItemSysId` - String that specifies the sys ID of the timeline item (a row) that the current span was dragged into.

elementMoveXY

public void elementMoveXY `(String spanSysId, String itemSysId, String newItemSysId, String newStartDateTimeMs)`

Event handler for when a timeline span moves both horizontally and vertically.

Parameters:

`spanSysId` - String that specifies the sys ID of the current span that is being adjusted.

`itemSysId` - String that specifies the sys ID of the timeline item associated with the current span.

`newItemSysId` - String that specifies the sys ID of the timeline item (a row) that the current span was dragged into.

`newStartDateTimeMs` - String that specifies the new start time of the span in milliseconds. Make sure to parse the string using `parseInt()` prior to performing any numerical comparisons.

elementSuccessor

public void elementSuccessor `(String spanSysId, String newSuccSpanId)`

Event handler for when a timeline relationship has been created between two spans.

Parameters:

`spanSysId` - String that specifies the sys ID of the current span which will be a predecessor for the newly created relationship.

`newSuccSpanId` - String that specifies the sys ID of the successor span from the relationship created.

elementTimeAdjustStart

public void elementTimeAdjustStart `(String spanSysId, String newStartDateTimeMs)`

Event handler for when a timeline span's start date was modified.

Parameters:

`spanSysId` - String that specifies the sys ID of the current span that is being adjusted.

`newStartDateTimeMs` - String that specifies the new start time of the span in milliseconds. Make sure to parse the string using `parseInt()` prior to performing any numerical comparisons.

elementTimeAdjustEnd

public void elementTimeAdjustEnd `(String spanSysId, String newStartDateTimeMs)`

Event handler for when a timeline span's end date was modified.

Parameters:

`spanSysId` - String that specifies the sys ID of the current span that is being adjusted.

`newStartDateTimeMs` - String that specifies the new start time of the span in milliseconds. Make sure to parse the string using `parseInt()` prior to performing any numerical comparisons.

inputBox

public void inputBox `(String strInputText)`

Event handler for when a string was typed into the left pane input box.

Parameters:

`strInputText` - String that specifies the text that was entered in the input box in the left pane

itemMove

public void itemMove `(String itemSysId, String newItemSysId)`

Event handler for when a timeline row item was moved and dragged into another row item.

Parameters:

`itemSysId` - String that specifies the sys ID of the timeline item associated with the current span.

`newItemSysId` - String that specifies the sys ID of the timeline item (a row) that the current span was dragged into.

Class GlideTimelineItem

This class extends the abstract `ScheduleItem` class to define additional properties that are specific to the timeline. A timeline item is essentially any item that is displayed in a singular row across the timeline. A `GlideTimelineItem` may exist where it has zero or more spans `(TimelineSpan` objects) associated with it.

See Also:

TimelineSpan

Constructor Summary

Return Value	Details
GlideTimelineItem	**GlideTimelineItem** `(String sys_id)` Non-default constructor that allows a "dummy" GlideTimelineItem to be created. Useful for creating rows that do not allow any YMoving into; however, contain nested children. (e.g. The top-level "Users" row in the Group Resource Timeline). The sys_id needs to be unique for DOM level functions to parse correctly.
GlideTimelineItem	**GlideTimelineItem** `(String table, String sys_id)` Constructor that sets the required table and sys_id properties. The rest of this object's properties should be set from the caller appropriately.

Method Summary

Return Value	Details
TimelineSpan	**createTimelineSpan** `(String tableName)` Creates a new `TimelineSpan` object associated with the current instance object. If no other `TimelineSpan` objects exist, the newly created object will share the same sys_id as current instance object. Otherwise, a randomly generated GUID will be used.
TimelineSpan	**createTimelineSpan** `(String tableName, String sysId)` Creates a new `TimelineSpan` object associated with the current instance object using the specified table and sysId.
String	**getImage** `()` Returns a string specifying the name of the image file associated with the current image. If no image is associated with the current image, an empty string ("") is returned.
Boolean	**getIsDroppable** `()` Returns a boolean that specifies whether or not the current instance object should be allowed as a "drop zone" when moving timeline elements vertically.
String	**getLeftLabelText** `()` Returns a string that specifies the text to display in the left pane (if enabled).
String	**getParent** `()` Returns the unique sysId of the current `TimelineItem`'s parent object. If the parent does not exist, this will return an empty string ("").
ArrayList	**getTimelineSpans** `()` Returns all the `TimelineSpan` objects associated with the current instance in an ArrayList.
Boolean	**isTextBold** `()` Returns a boolean that specifies if the left pane text should be displayed using a **bold** style.
void	**setImage** `(String strImageName)` Specifies the name of the image file (including it's path) to use as the icon for the item in the left pane.
void	**setIsDraggable** `(Boolean b)` Specifies whether or not the current instance object can be clicked and dragged into another Timeline Item.
void	**setIsDroppable** `(Boolean b)` Specifies whether or not the current instance object should be allowed as a "drop zone" when moving timeline elements vertically.
void	**setLeftLabelText** `(String strText)` Specifies the text to display in the left pane for this item.
void	**setParent** `(String parentSysId)` Specifies the sys ID of another `TimelineItem` who is a parent of the current instance.

void	**setTextBold** `(Boolean b)` Specifies whether or not to bold the text style of the item in the left pane.

Back to Application Programming Interface (API)

GlideTimelineItem

public GlideTimelineItem `(String tableName)`

Non-default constructor that allows a "dummy" GlideTimelineItem to be created. Useful for creating rows that do not allow any YMoving into; however, contain nested children. (e.g. The top-level "Users" row in the Group Resource Timeline). The sys_id needs to be unique for DOM level functions to parse correctly. By default this object will **not** be "droppable" because a table name was not specified.

Parameters:

`tableName` - String that specifies the name of the table that is associated with current object.

GlideTimelineItem

public GlideTimelineItem `(String tableName, String sys_id)`

Constructor that sets the required table and sys_id properties. The rest of this object's properties should be set from the caller appropriately. By default, this object instance is "droppable" since a table name was specified.

Parameters:

`tableName` - String that specifies the name of the table that is associated with current object.

`sys_id` - String that specifies the sys ID for the object.

createTimelineSpan

public TimelineSpan createTimelineSpan `(String tableName)`

Creates a new `TimelineSpan` object associated with the current instance object. If no other `TimelineSpan` objects exist, the newly created object will share the same sys_id as current instance object. Otherwise, a randomly generated GUID will be used.

Parameters:

`tableName` - String that specifies the name of the table that is associated with current object.

Returns:

`TimelineSpan` - The newly created span object instance.

createTimelineSpan

public TimelineSpan createTimelineSpan `(String tableName, String sysId)`

Creates a new TimelineSpan object associated with the current instance object using the specified table and sysId.

Parameters:

`tableName` - String that specifies the name of the table that is associated with current object.

`sys_id` - String that specifies the sys ID for the object.

Returns:

`TimelineSpan` - The newly created span object instance.

getImage

public String getImage()

Returns a string specifying the name of the image file associated with the current image. If no image is associated with the current image, an empty string ("") is returned.

Returns:

`String` - The String name of the image file and path if specified; otherwise "".

getIsDroppable

public Boolean getIsDroppable()

Returns a boolean that specifies whether or not the current instance object should be allowed as a "drop zone" when moving timeline elements vertically.

Returns:

`Boolean` - `true` if droppable; `false` otherwise.

getLeftLabelText

public String getLeftLabelText()

Returns a string that specifies the text to display in the left pane (if enabled).

Returns:

`String` - The String value of the left label text if specified; otherwise "".

getParent

public String getParent()

Returns the unique sysId of the current TimelineItem's parent object. If the parent does not exist, this will return an empty string ("").

Returns:

`String` - The String that specifies the parent `TimelineItem` of the current instance if specified; otherwise "".

getTimelineSpans

public ArrayList getTimelineSpans()

Returns all the TimelineSpan objects associated with the current instance in an ArrayList.

Returns:

`ArrayList` - The list of `TimelineSpan` objects associated with the current instance.

isTextBold

public Boolean isTextBold()

Returns a boolean that specifies if the left pane text should be displayed using a bold style.

Returns:

`Boolean` - `true` if the text should be **bolded**; otherwise `false`.

setImage

public void setImage (`String strImageName`)

Specifies the name of the image file (including it's path) to use as the icon for the item in the left pane.

Parameters:

`strImageName` - The name of the image including it's path.

setIsDraggable

public void setIsDraggable (`Boolean b`)

Specifies whether or not the current instance object can be clicked and dragged into another Timeline Item.

Parameters:

`b` - `true` if the item can be clicked and moved; otherwise, `false`.

setIsDroppable

public void setIsDroppable (`Boolean b`)

Specifies the name of the image file (including it's path) to use as the icon for the item in the left pane.

Parameters:

`b` - `true` if the item should be marked as "droppable"; otherwise, `false`.

setLeftLabelText

public void setLeftLabelText (`String strText`)

Specifies the text to display in the left pane for this item.

Parameters:

`strText` - The text to display in the left pane for this item.

setParent

public void setParent (`String parentSysId`)

Specifies the sys ID of another `TimelineItem` who is a parent of the current instance.

Parameters:

`parentSysId` - Parent `TimelineItem` sys Id.

setTextBold

public void setTextBold (`Boolean b`)

Specifies whether or not to bold the text style of the item in the left pane.

Parameters:

`b` - `true` if the style of the left pane item text should be **bold**; otherwise `false`.

Class TimelineSpan

This class defines a set of properties that describe the characteristics and interactive behavior of an element rendered within a `GlideTimelineItem`. Since it is important for all of a `GlideTimelineItem`'s collection of spans to be unique, the creation of a new instance should be performed via the `createTimelineItem` method of an existing `GlideTimelineItem` instance.

See Also:

GlideTimelineItem

Method Summary	
Return Value	**Details**
void	**addPredecessor** `(Object[] objArray)` Adds multiple relationships between the current instance and other `TimelineSpan` objects by enumerating through the array of JavaScript objects. Each object should have an internal property "relationship_sys_id" and "predecessor_sys_id" specified.
void	**addPredecessor** `(String strPredecessorSysId, String strRelationshipSysId)` Adds the specified relationship between the current instance and another `TimelineSpan` with sys ID `strPredecessorSysId`. The drawn line will not have any double click handlers associated with it.
void	**addPredecessor** `(String strPredecessorSysId, String strRelationshipSysId, String strTableName)` Adds the specified relationship between the current instance and another `TimelineSpan` with sys ID `strPredecessorSysId` and allow the relationship to open a GlideWindow to display information about the relationship.
Boolean	**getAllowXDragLeft** `()` Returns the boolean value of the AllowXDragLeft property.
Boolean	**getAllowXDragRight** `()` Returns the boolean value of the AllowXDragRight property.
Boolean	**getAllowXMove** `()` Returns the boolean value of the AllowXMove property.
Boolean	**getAllowYMove** `()` Returns the boolean value of the AllowYMove property.
Boolean	**getAllowYMovePredecessor** `()` Returns the boolean value of the AllowYMovePredecessor property.
Number	**getEndTimeMs** `()` Returns the start time in milliseconds of the current instance object as a long Number.
String	**getInnerSegmentClass** `()` Returns a String that specifies the current inner segment class for the Timeline Span.
Number	**getInnerSegmentEndTimeMs** `()` Returns the time in milliseconds of the end time of the inner segment portion of the timeline span as a long Number.
Number	**getInnerSegmentStartTimeMs** `()` Returns the time in milliseconds of the start time of the inner segment portion of the timeline span as a long Number.
Boolean	**getIsChanged** `()` Returns a boolean that specifies whether or not the current timeline item has been modified after initialization.
String	**getPointIconClass** `()` Returns a String that specifies the name of the icon class to use for displaying the element on the timeline if the current instance has zero duration.

ArrayList	**getPredecessors** `()` Returns an ArrayList of all the predecessor objects associated with the current instance. Each ArrayList object is a HashMap that contains a `predecessor_sys_id` and `relationship_sys_id` property.
String	**getSpanColor** `()` Returns the string name of the color specified for displaying this span.
String	**getSpanText** `()` Returns the string that specifies the text to display adjacent to the time element. Note that this text will only be displayed if the GlideTimeline object has enabled timeline text via `glideTimeline.showTimelineText(true).`
Number	**getStartTimeMs** `()` Returns the start time in milliseconds of the current instance object as a long Number.
String	**getSysId** `()` Returns the sys ID of the current object. This method is useful for returning the sys Id when the current object instance was created without a specific sys Id to obtain the dynamically generated GUID.
String	**getTable** `()` Returns the string name of the table where the sys ID is referenced.
String	**getTooltip** `()` Returns the string that specifies the text/html to display in the tooltip when the TimeSpan element is being hovered over.
void	**setAllowXDragLeft** `(Boolean b)` Sets a flag that determines whether the element's start date can be dragged left or right therefore adjusting the duration of the task. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.
void	**setAllowXDragRight** `(Boolean b)` Sets a flag that determines whether the element's end date can be dragged left or right therefore adjusting the duration of the task. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.
void	**setAllowXMove** `(Boolean b)` Sets a flag that determines whether the element can be moved to start at a different time. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.
void	**setAllowYMove** `(Boolean b)` Sets a flag that determines whether the element can be dragged vertically on the timeline. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.
void	**setAllowYMovePredecessor** `(Boolean b)` Sets a flag that determines whether a dashed relationship line can be drawn from this element interactively on the timeline. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.
void	**setInnerSegmentClass** `(String s)` Specifies the name of the class to use for stylizing the inner segment if it exists.
void	**setInnerSegmentTimeSpan** `(String startTimeMs, String endTimeMs)` Creates an inner segment to show within the current timespan defined by the range specified.
void	**setInnerSegmentTimeSpan** `(Number startTimeMs, Number endTimeMs)` Creates an inner segment to show within the current timespan defined by the range specified.
void	**setIsChanged** `(Boolean b)` Sets a flag that specifies whether or not the current timeline item has been modified after initialization. The default value for this property is `false`.

void	**setPointIconClass** `(String s)` Sets the icon class to use for displaying the current element on the timeline if the current instance has zero duration. Note that this only affects the current `TimelineSpan` object and will take precedence over the defaultPointIconClass specified by the `GlideTimeline`.
void	**setSpanColor** `(String s)` Sets the color for displaying this span. This needs to be a valid HTML color entity specified by a named color or a hexadecimal color code.
void	**setSpanText** `(String s)` Sets the text to display adjacent to the time element. Note that this text will only be displayed if the GlideTimeline object has enabled timeline text via `glideTimeline.showTimelineText(true)`.
void	**setTimeSpan** `(long startTimeMs, long endTimeMs)` Sets the start and end dates for the current span.
void	**setTimeSpan** `(String startTimeMs, String endTimeMs)` Sets the start and end dates for the current span.
void	**setTooltip** `(String s)` Sets the text to display in the tooltip when the TimeSpan element is being hovered over. **Note:** *You can specify valid HTML in the string that sets the tooltip.*

Back to Application Programming Interface (API)

addPredecessor

public void addPredecessor `(Object[] objArray)`

Adds multiple relationships between the current instance and other `TimelineSpan` objects by enumerating through the array of JavaScript objects. Each object should have an internal property "**relationship_sys_id**" and "**predecessor_sys_id**" specified.

Parameters:

`objArray` - JavaScript object array that contains two internal properties: **relationship_sys_id** and **predecessor_sys_id**.

addPredecessor

public void addPredecessor `(String strPredecessorSysId, String strRelationshipSysId)`

Adds the specified relationship between the current instance and another `TimelineSpan` with sys ID `strPredecessorSysId`. The drawn line will not have any double click handlers associated with it.

Parameters:

`strPredecessorSysId` - String that specifies the sys ID of the planned task that is the predecessor of the relationship.

`strRelationshipSysId` - String that specifies the sys ID of the relationship of the relationship.

addPredecessor

public void addPredecessor`(String strPredecessorSysId, String strRelationshipSysId, String strTableName)`

Adds the specified relationship between the current instance and another `TimelineSpan` with sys ID `strPredecessorSysId` and allow the relationship to open a GlideWindow to display information about the relationship.

Parameters:

`strPredecessorSysId` - String that specifies the sys ID of the planned task that is the predecessor of the relationship.

`strRelationshipSysId` - String that specifies the sys ID of the relationship of the relationship.

`strTableName` - String that specifies the name of the table for the relationship.

getAllowXDragLeft

public Boolean getAllowXDragLeft`()`

Returns the boolean value of the AllowXDragLeft property.

Returns:

`Boolean` - `true` if the object's start time can be adjusted; `false` otherwise.

getAllowXDragRight

public Boolean getAllowXDragRight`()`

Returns the boolean value of the AllowXDragRight property.

Returns:

`Boolean` - `true` if the object's end time can be adjusted; `false` otherwise.

getAllowXMove

public Boolean getAllowXMove`()`

Returns the boolean value of the AllowXMove property.

Returns:

`Boolean` - `true` if the object can be moved; `false` otherwise.

getAllowYMove

public Boolean getAllowYMove`()`

Returns the boolean value of the AllowYMove property.

Returns:

`Boolean` - `true` if the object can be moved vertically; `false` otherwise.

getAllowYMovePredecessor

public Boolean getAllowYMovePredecessor ()

Returns the boolean value of the AllowYMovePredecessor property.

Returns:

`Boolean` - `true` if the a dashed relationship line can be drawn from the current object to a new successor; `false` otherwise.

getEndTimeMs

public Number getEndTimeMs ()

Returns the start time in milliseconds of the current instance object as a long Number.

Returns:

`Number` - The end time of the timeline span in milliseconds.

getInnerSegmentClass

public String getInnerSegmentClass ()

Returns a String that specifies the current inner segment class for the Timeline Span.

Returns:

`String` - The String name of the class for the current inner segment style.

getInnerSegmentEndTimeMs

public Number getInnerSegmentEndTimeMs ()

Returns the time in milliseconds of the end time of the inner segment portion of the timeline span as a long Number.

Returns:

`Number` - The end time of the timeline span inner segment portion in milliseconds.

getInnerSegmentStartTimeMs

public Number getInnerSegmentStartTimeMs ()

Returns the time in milliseconds of the start time of the inner segment portion of the timeline span as a long Number.

Returns:

`Number` - The start time of the timeline span inner segment portion in milliseconds.

getIsChanged

public Boolean getIsChanged`()`

Returns a boolean that specifies whether or not the current timeline item has been modified after initialization.

Returns:

`Boolean` - `true` if the current span has been marked as **changed**; otherwise `false`.

getPointIconClass

public String getPointIconClass`()`

Returns a String that specifies the name of the icon class to use for displaying the element on the timeline if the current instance has zero duration.

Returns:

`String` - The name of the icon class to use for displaying the current `TimelineSpan` if the duration is zero.

getPredecessors

public ArrayList getPredecessors`()`

Returns an ArrayList of all the predecessor objects associated with the current instance. Each ArrayList object is a HashMap that contains a **predecessor_sys_id** and **relationship_sys_id** property.

Returns:

`ArrayList` - List of HashMap's that contain two internal properties: **predecessor_sys_id** and **relationship_sys_id**.

getSpanColor

public String getSpanColor`()`

Returns the string name of the color specified for displaying this span.

Returns:

`String` - String that specifies the HTML color name to use as the background color for the element.

getSpanText

public String getSpanText`()`

Returns the string that specifies the text to display adjacent to the time element.

Note: *Note that this text will only be displayed if the GlideTimeline object has enabled timeline text via* `glideTimeline.showTimelineText(true)`*.*

Returns:

`String` - String value that contains the text to display adjacent to the element.

getStartTimeMs

public Number getStartTimeMs ()

Returns the start time in milliseconds of the current instance object as a long Number.

Returns:

`Number` - String that specifies the start time of the element in miliseconds.

getSysId

public String getSysId ()

Returns the sys ID of the current object. This method is useful for returning the sys Id when the current object instance was created without a specific sys Id to obtain the dynamically generated GUID.

Returns:

`String` - String that specifies the unique system ID of the current element.

getTable

public String getTable ()

Returns the string name of the table where the sys ID is referenced.

Returns:

`String` - String that specifies the table name.

getTooltip

public String getTooltip ()

Returns the string that specifies the text/html to display in the tooltip when the TimeSpan element is being hovered over.

Returns:

`String` - String that specifies the tooltip text.

setAllowXDragLeft

public void setAllowXDragLeft (Boolean b)

Sets a flag that determines whether the element's start date can be dragged left or right therefore adjusting the duration of the task. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.

Parameters:

`b` - `true` if the element's start date can be adjusted; `false` otherwise.

setAllowXDragRight

public void setAllowXDragRight `(Boolean b)`

Sets a flag that determines whether the element's end date can be dragged left or right therefore adjusting the duration of the task. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.

Parameters:

`b` - `true` if the element's end date can be adjusted; `false` otherwise.

setAllowXMove

public void setAllowXMove `(Boolean b)`

Sets a flag that determines whether the element can be moved to start at a different time. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.

Parameters:

`b` - `true` if the element can be moved horizontally; `false` otherwise.

setAllowYMove

public void setAllowYMove `(Boolean b)`

Sets a flag that determines whether the element can be dragged vertically on the timeline. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.

Parameters:

`b` - `true` if the element can be moved vertically; `false` otherwise.

setAllowYMovePredecessor

public void setAllowYMovePredecessor `(Boolean b)`

Sets a flag that determines whether a dashed relationship line can be drawn from this element interactively on the timeline. The effect of this behavior is controlled by the script include that handles the appropriate event. The default value for this property is `false`.

Parameters:

`b` - `true` if a relationship line can be drawn **from** this element; `false` otherwise.

setInnerSegmentClass

public void setInnerSegmentClass `(String s)`

Specifies the name of the class to use for stylizing the inner segment if it exists. The default value is `green`.

Parameters:

`s` - String that specifies one of the following values:

- `green` 🟩
- `blue` 🟦
- `silver` ⬜

setInnerSegmentTimeSpan

public void setInnerSegmentTimeSpan `(String startTimeMs, String endTimeMs)`

Creates an inner segment to show within the current timespan defined by the range specified.

Parameters:

`startTimeMs` - String that specifies the start time in milliseconds.

`endTimeMs` - String that specifies the end time in milliseconds.

setInnerSegmentTimeSpan

public void setInnerSegmentTimeSpan `(Number startTimeMs, Number endTimeMs)`

Creates an inner segment to show within the current timespan defined by the range specified.

Parameters:

`startTimeMs` - Number that specifies the start time in milliseconds.

`endTimeMs` - Number that specifies the end time in milliseconds.

setIsChanged

public void setIsChanged `(Boolean b)`

Sets a flag that specifies whether or not the current timeline item has been modified after initialization. The default value for this property is `false`.

Parameters:

`b` - `true` if the current element is marked as **changed**; `false` otherwise.

setPointIconClass

public void setPointIconClass `(String s)`

Sets the icon class to use for displaying the current element on the timeline if the current instance has zero duration. Note that this only affects the current `TimelineSpan` object and will take precedence over the defaultPointIconClass specified by the `GlideTimeline`.

Parameters:

`s` - String that specifies one of the following values:

- `milestone` ◆
- `blue_square` 🟦
- `sepia_square` 🟫
- `green_square` 🟩
- `red_square` 🟥
- `black_square` ⬛
- `blue_circle` 🔵
- `sepia_circle` 🟤
- `green_circle` 🟢
- `red_circle` 🔴
- `black_circle` ⚫

setSpanColor

public void setSpanColor(`String s`)

Sets a flag that specifies whether or not the current timeline item has been modified after initialization. The default value for this property is `false`.

Parameters:

`s` - String that specifies the HTML color code.

setSpanText

public void setSpanText(`String s`)

Sets the text to display adjacent to the time element. Note that this text will only be displayed if the GlideTimeline object has enabled timeline text via `glideTimeline.showTimelineText(true)`.

Parameters:

`s` - String that specifies the description text.

setTimeSpan

public void setTimeSpan(`long startTimeMs, long endTimeMs`)

Sets the start and end dates for the current span.

Parameters:

`startTimeMs` - A number that specifies the start time in milliseconds.

`endTimeMs` - A number that specifies the end time in milliseconds.

setTimeSpan

public void setTimeSpan(`String startTimeMs, String endTimeMs`)

Sets the start and end dates for the current span.

Parameters:

`startTimeMs` - A string that specifies the start time in milliseconds.

`endTimeMs` - A string that specifies the end time in milliseconds.

setTooltip

public void setTooltip(`String s`)

Sets the text to display in the tooltip when the TimeSpan element is being hovered over.

Note: *You can specify valid HTML in the string that sets the tooltip.*

Parameters:

`s` - A string that specifies the tooltip text.

Example

The following example demonstrates how to create a timeline schedule page with corresponding script include utilizing a majority of the API described above. For this example we are going to create an Incident Summary Timeline for a project support manager to visualize all of the new incidents. All new incidents should be displayed as single points where the priority of the incident is distinguished by a different point icon. Additionally, all closed incidents should be displayed on the timeline in a separate group that shows the duration of the incident before it was closed. Since the Project Manager wants to be able to easily close new items that are resolved without using any form lists, we will handle the vertical move event allowing the new incidents to be dragged into the closed incident group or items within.

Schedule Page

Create a new Schedule Page with the following properties:

- **Name:** Hardware Incidents
- **Schedule type:** incident_timeline
- **View Type:** Timeline
- **Client Script:**

```
// Set our page configuration
glideTimeline.setReadOnly(false);
glideTimeline.showLeftPane(true);
glideTimeline.showLeftPaneAsTree(true);
glideTimeline.showTimelineText(true);
glideTimeline.showDependencyLines(false);
glideTimeline.groupByParent(true);
glideTimeline.setDefaultPointIconClass('milestone');

// We will define what items to display and provide a custom event
handler for moving new items to the closed state
glideTimeline.registerEvent('getItems',
'IncidentTimelineScriptInclude');
glideTimeline.registerEvent('elementMoveY',
'IncidentTimelineScriptInclude');
```

Script Include

Now that the schedule page has been created we need to generate a matching script include for the events that were registered. Create a new script include with the following properties:

- **Name:** IncidentTimelineScriptInclude
- **Active:** Checked
- **Client callable:** Checked
- **Script:**

```
// Class Imports

var IncidentTimelineScriptInclude = Class.create();
IncidentTimelineScriptInclude.prototype =
Object.extendsObject(AbstractTimelineSchedulePage, {
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
  // GET_ITEMS

////////////////////////////////////////////////////////////////////////////////////////////////
  getItems: function() {
    // Specify the page title
    this.setPageTitle('My Custom Incident Summary Timeline');

    var groupNew = new GlideTimelineItem('new');
    groupNew.setLeftLabelText('New Incidents');
    groupNew.setImage('../images/icons/all.gifx');
    groupNew.setTextBold(true);
    this.add(groupNew);

    var groupClosed = new GlideTimelineItem('closed');
    groupClosed.setLeftLabelText('Closed Incidents');
    groupClosed.setImage('../images/icons/all.gifx');
    groupClosed.setTextBold(true);
    groupClosed.setIsDroppable(true); // This allows us to drag an open
 incident onto the closed group row.
    this.add(groupClosed);

    // Get all the incidents and let's add only the new/closed ones
appropriately
    var gr = new GlideRecord('incident');
    gr.query();
    while (gr.next()) {
      // Only loop through new/closed incidents
      if (gr.incident_state != '1' && gr.incident_state != '7')
        continue;

      // Ok, we have a new/closed incident. Create the item and the
span first.
      var item = new GlideTimelineItem(gr.getTableName(), gr.sys_id);
      var span = item.createTimelineSpan(gr.getTableName(), gr.sys_id);

      // Specific properties for a new incident
      if (gr.incident_state == '1') { // New
        item.setParent(groupNew.getSysId());
        item.setImage('../images/icons/open.gifx');

span.setTimeSpan(gr.getElement('opened_at').getGlideObject().getNumericValue(),

gr.getElement('opened_at').getGlideObject().getNumericValue());
```

```
        // We'll show different colors based upon the priorities only
for new incidents
        switch (gr.getElement('priority').toString()) {
          case '1':
            span.setPointIconClass('red_circle');
            break;
          case '2':
            span.setPointIconClass('red_square');
            break;
          case '3':
            span.setPointIconClass('blue_circle');
            break;
          case '4':
            span.setPointIconClass('blue_square');
            break;
          case '5':
            span.setPointIconClass('sepia_circle');
            break;
          default: // Otherwise, the default point icon class will be
used (Milestone)
        }

      // Specific properties for a closed incident
      } else if (gr.incident_state == '7') {
        item.setParent(groupClosed.getSysId());
        item.setImage('../images/icons/closed.gifx');

span.setTimeSpan(gr.getElement('opened_at').getGlideObject().getNumericValue(),

gr.getElement('closed_at').getGlideObject().getNumericValue());
      }

      // Common item properties
      item.setLeftLabelText(gr.short_description);

      // Common span properties
      span.setSpanText(gr.short_description);
      span.setTooltip('<strong>' +
GlideStringUtil.escapeHTML(gr.short_description) + '</strong><br>' + gr.number);
      span.setAllowXMove(false);
      span.setAllowYMove(gr.canWrite() ? true : false);
      span.setAllowYMovePredecessor(false);
      span.setAllowXDragLeft(false);
      span.setAllowXDragRight(false);

      // Now we add the actual item
      this.add(item);
```

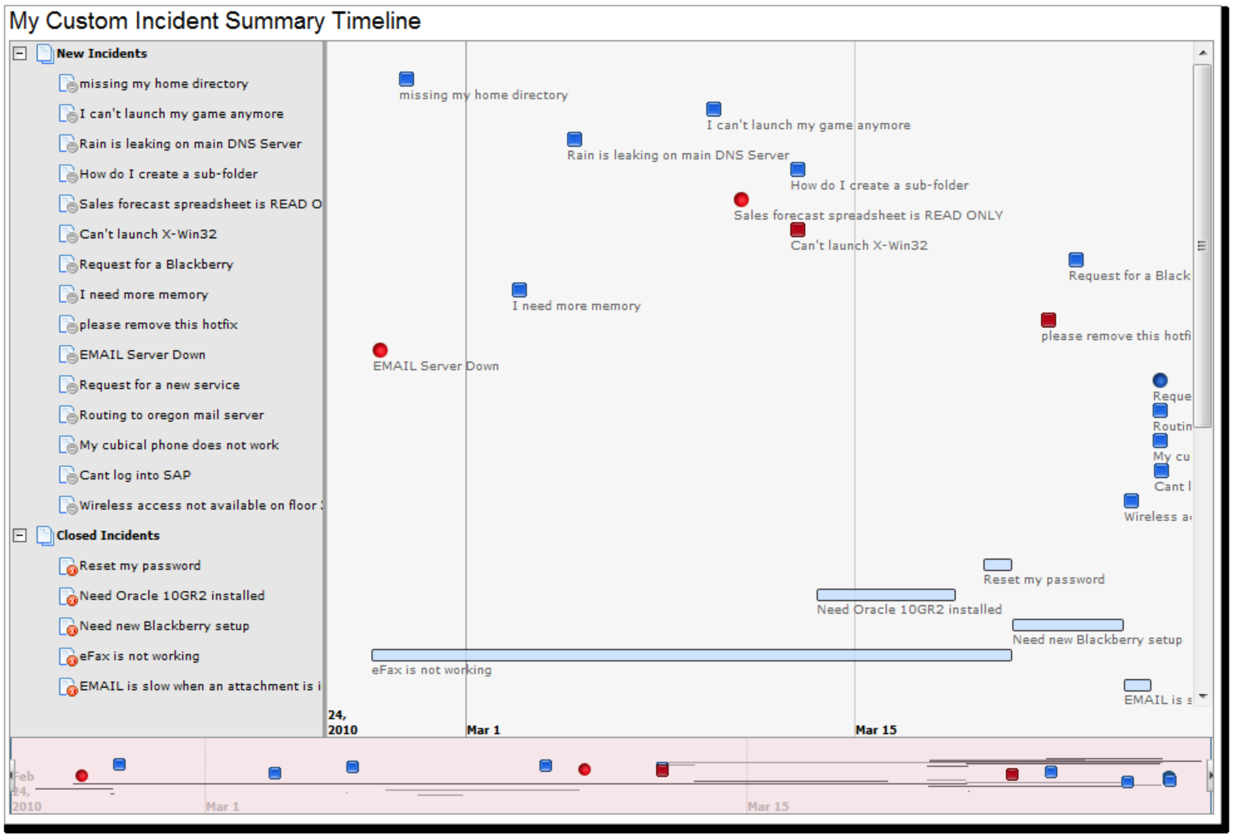
```
    }
  },




///////////////////////////////////////////////////////////////////////////////////////////////
  // ELEMENT_MOVE_Y

///////////////////////////////////////////////////////////////////////////////////////////////

  /**
   * This is one of the AbstractTimelineSchedulePage event handler
methods that corresponds to a vertical
   * move. The arguments for this function are defined in the API
section of the event handler methods.
   */
  elementMoveY: function(spanId, itemId, newItemId) {

    // Get information about the current incident
    var gr = new GlideRecord('incident');
    gr.addQuery('sys_id', spanId);
    gr.query();
    if (!gr.next())
      return this.setStatusError('Error', 'Unable to lookup the current
 incident.');

    // Only allow the new incidents to have their state adjusted.
    if (gr.incident_state != '1')
      return this.setStatusError('Error', 'Only new incidents can have
their state adjusted.');

    // Get information about the dropped TimelineItem. If it was
dropped in an item on the "New Incidents"
    // group let's do nothing. If it was dropped in the "Closed
Incidents" then let's adjust the state automatically.
    var grDropped = new GlideRecord('incident');
    grDropped.addQuery('sys_id', newItemId);
    grDropped.query();
    if (!grDropped.next() || grDropped.incident_state == '7') {
      // This means the dropped item was either the 'Closed Incidents'
group (which has no record or sys_id) or an
      // existing incident that is closed. The 'New Incidents' also has
 no sys_id; however, the default behavior for
      // items without a sysId is to be non-droppable. This is why we
explicitly denoted the 'Closed Incidents' to
      // be marked as "droppable".
      // Return a dialog prompt
      this.setStatusPrompt('Confirm',
        'Are you sure you want to close: ' +
          '<div style="margin:10px 0 10px 14px;padding:4px;background-color:#EBEBEB;"><strong>' +
          GlideStringUtil.escapeHTML(gr.short_description) +
          '</strong><br /><div class="font_smaller">' + gr.number + '</div></div>',
        'this._elementMoveYHandler_DoClose',      // This function is
for when the OK button is clicked.
        'this._elementMoveYHandler_DoNothing',    // This function is
for when the Cancel button is clicked.
        'this._elementMoveYHandler_DoNothing');   // This function is
for when the Close button is clicked.
    }
  },

  _elementMoveYHandler_DoClose: function(spanId, itemId, newItemId) {
    // Notice that this function takes the same function arguments as
the original function for which it
    // is a custom event handler for.

    // Update the database record from 'New' to 'Closed'.
    var gr = new GlideRecord('incident');
    gr.addQuery('sys_id', spanId);
    gr.query();
    gr.next();
    gr.setValue('incident_state', '7');
    gr.update();

    // This will re-render the timeline showing the updated item in the
 closed group.
    this.setDoReRenderTimeline(true);

    // Let's show a success message
    this.setStatusSuccess('Success', '<strong>' + gr.short_description + '</strong> was
successfully closed.');
  },

  // Since the user clicked cancel or close we simply do nothing.
  _elementMoveYHandler_DoNothing: function(spanId, itemId, newItemId)
{}

});
```

Screenshots / Results

1. Navigate to:

   http://[YOURINSTANCE]:8080/show_schedule_page.do?sysparm_page_schedule_type=**incident_timeline**

   Notice the bold text is the value of the schedule page **Schedule type** field.
2. The page displays a timeline as specified by the schedule page and script include created. A link to this page can be created and placed as a Module or UI Action as necessary.

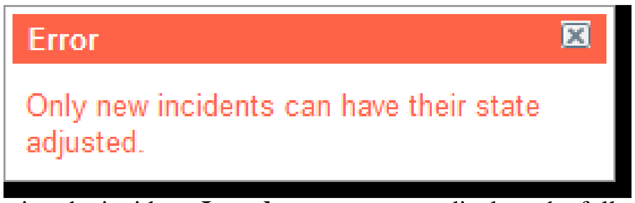


My Custom Incident Summary Timeline

New Incidents
- missing my home directory
- I can't launch my game anymore
- Rain is leaking on main DNS Server
- How do I create a sub-folder
- Sales forecast spreadsheet is READ O
- Can't launch X-Win32
- Request for a Blackberry
- I need more memory
- please remove this hotfix
- EMAIL Server Down
- Request for a new service
- Routing to oregon mail server
- My cubical phone does not work
- Cant log into SAP
- Wireless access not available on floor :

Closed Incidents
- Reset my password
- Need Oracle 10GR2 installed
- Need new Blackberry setup
- eFax is not working
- EMAIL is slow when an attachment is i

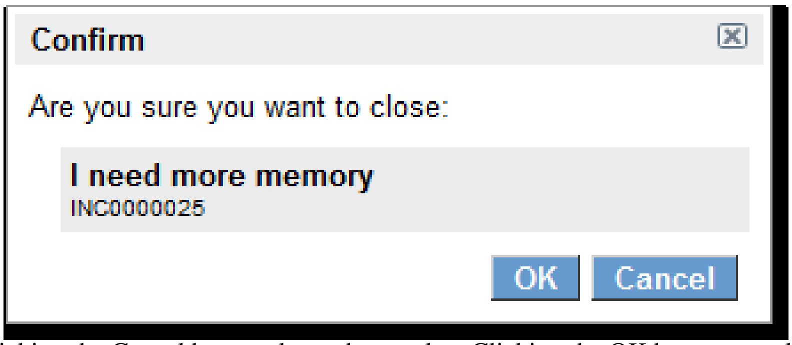
3. Attempting to move a closed incident anywhere displays the expected error message.



Error

Only new incidents can have their state adjusted.

4. Moving the incident: **I need more memory** displays the following confirmation box.

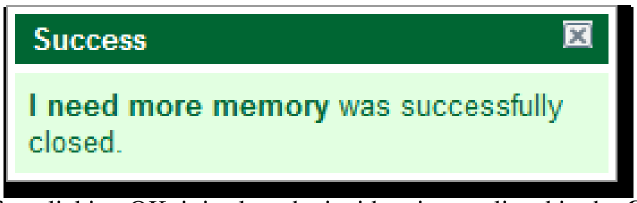


Confirm

Are you sure you want to close:

I need more memory
INC0000025

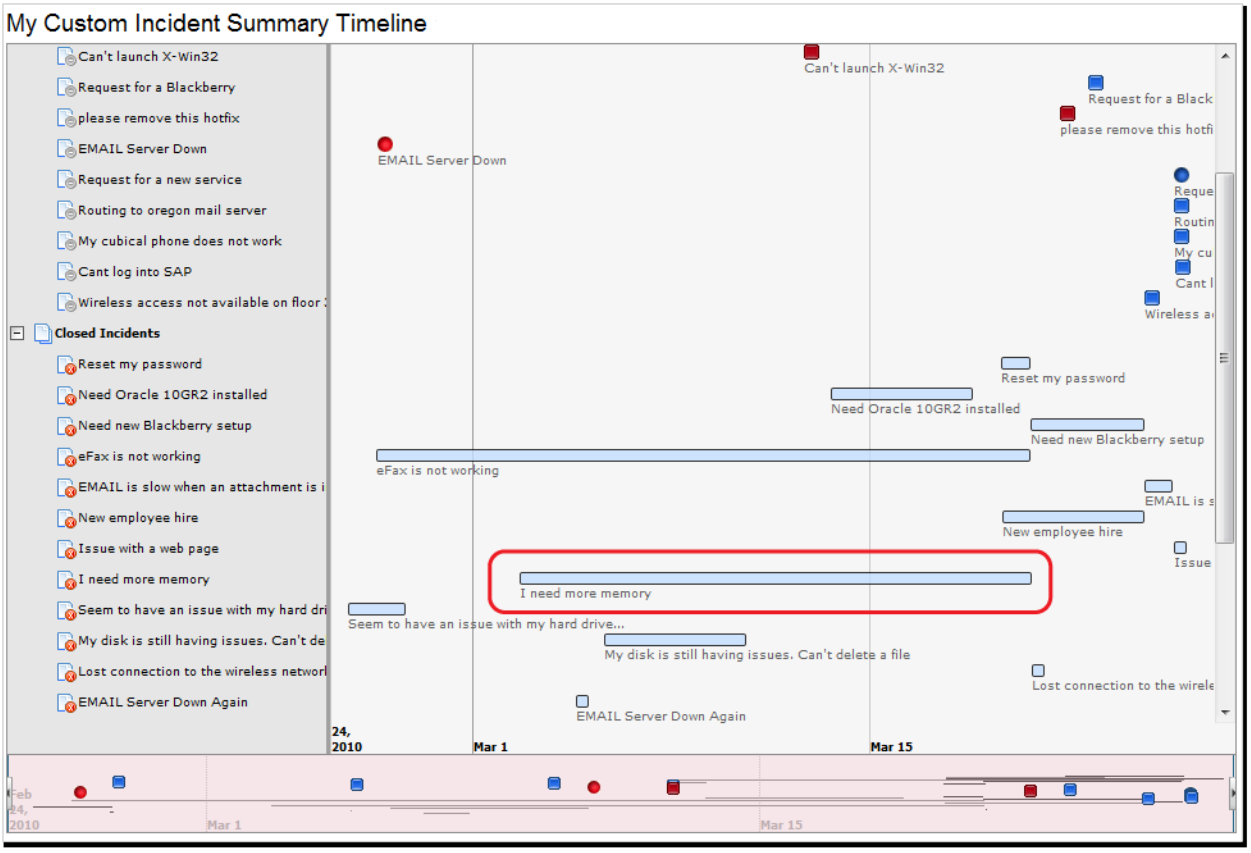
5. Clicking the Cancel button closes the overlay. Clicking the OK button actually updates the incident_state of the record and then displays the following success box.



Success

I need more memory was successfully closed.

6. After clicking OK, it is clear the incident is now listed in the **Closed Incidents** group.



My Custom Incident Summary Timeline

Can't launch X-Win32
Request for a Blackberry
please remove this hotfix
EMAIL Server Down
Request for a new service
Routing to oregon mail server
My cubical phone does not work
Cant log into SAP
Wireless access not available on floor
Closed Incidents
Reset my password
Need Oracle 10GR2 installed
Need new Blackberry setup
eFax is not working
EMAIL is slow when an attachment is i
New employee hire
Issue with a web page
I need more memory
Seem to have an issue with my hard dri
My disk is still having issues. Can't de
Lost connection to the wireless networl
EMAIL Server Down Again

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/script/server-api/concept/c_SchedulePages.html

Timeline Pages

Overview

Use timeline pages to track any activity bounded by two dates, such as the start and end date of a change request or the open and close date of an incident. For details on timeline navigation, filtering, and working with calendar perspectives, see Using Timelines.

Features

- Make selected timelines available to users by role.
- Select perspective from daily to yearly views.
- Create dynamic labels for timeline spans.
- Configure tooltips for each span.
- Permit span dragging and resizing by users.
- Lock timelines to prevent editing.

Creating a Timeline Page

To create a new timeline page:

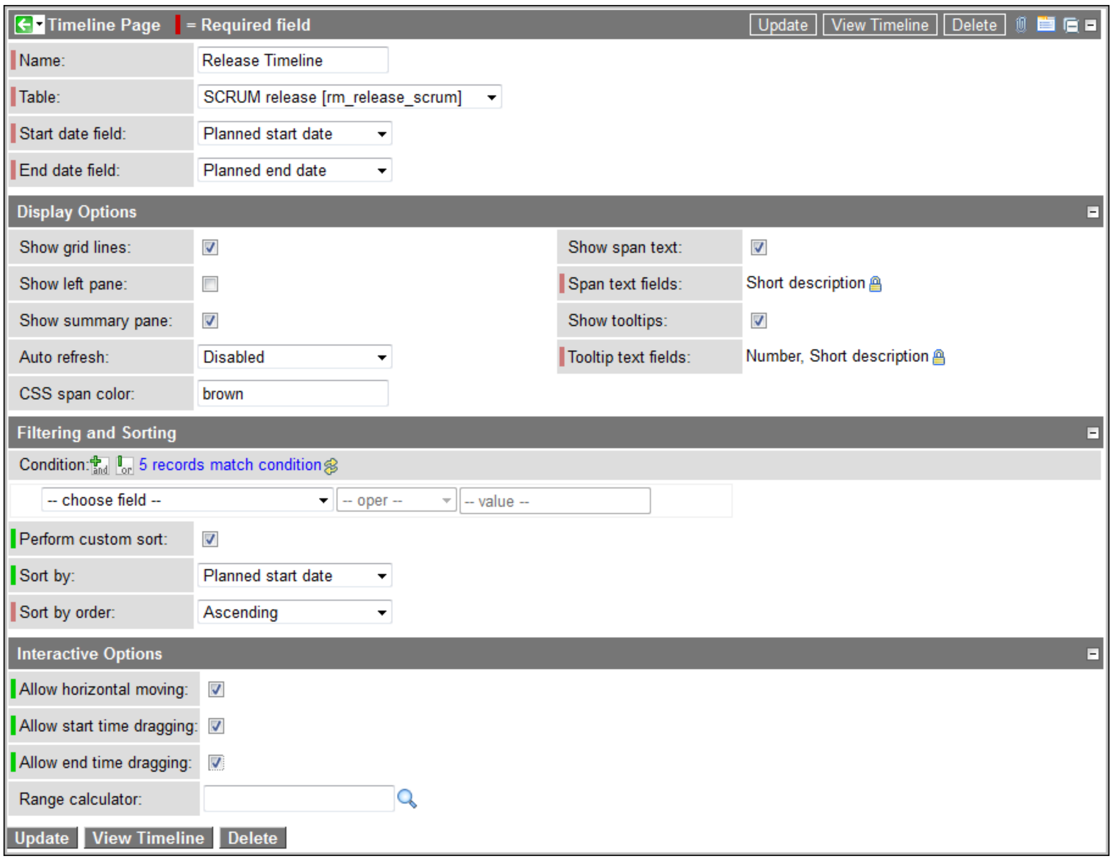
1. Navigate to **System UI > Timeline Pages**.
2. Click **New**.
3. Fill in the form as described in the table and click **Submit**.

The Timeline Page form provides the following fields.

Field	Description
Name	Enter a unique name that describes the function of this timeline. For example, **High Priority Change Requests**.
Table	Select the name of the table associated with this timeline, such as **Change Request [change_request]**. **Note:** *The list shows only tables and database views that are in the same scope as the timeline (starting with the Fuji release).*
Start date field	Select a time-related field from the specified table to use as the start date for the timeline. The timeline begins with the span for the record with the earliest start date from this field, after the filter and sort order are applied. For example, you can select **Updated** as the start date field and start the span for each active change request on the date it was updated to a high priority.
End date field	Select a time-related field from the specified table to use as the end date for the timeline. The timeline ends with the span for the record with the latest date from this field, after the filter and sort order are applied. For example, you can select **Closed** as the end date field and display all high priority change requests by the date on which they were closed.
Display Options	
Show grid lines?	Select this check box to show horizontal background shading to highlight alternate spans.
Show left pane?	Select this check box to show label text in a pane on the left of the timeline. The text that appears in this pane is defined in **Span text fields**.
Show summary pane	Select this check box to show the pink, perspective slider at the bottom of the timeline. Move the slider from right to left to scroll across the chart. Adjust the end points of the slider to change the magnification. A narrow slider zooms in on the spans and provides a more detailed view. A wide slider pulls the view out and makes more of the timeline visible on the screen.

Auto refresh	Select an automatic refresh interval or disable automatic refresh. When auto refresh is disabled, the timeline adjusts only when the browser is manually refreshed or when a start or end date field is updated in a record.
CSS span color	Enter a custom span color using any CSS color format, such as RGB or hexadecimal. If this field is blank, the default span color, light blue, is used.
Show span text	Select this check box to display the content of the **Span text fields** as labels below each span.
Span text fields	Select fields from the specified table to have those values appear as span labels. For example, you might select **Number** and **Short description**. The span labels also appear in the left pane if the left pane is visible.
Show tooltips	Select this check box to display tooltips when the cursor rests on a span.
Tooltip text fields	Select from the specified table the fields whose values appear as tooltips. For example, you might select **Category**, **Assigned to**, and **Due date**.
Filtering and Sorting	
Condition	Create a condition to filter the results that appear in the timeline. For example, a condition that displays only active, high priority incidents. This field has a Condition Count Widget to preview what records will be returned by this condition set.
Perform custom sort?	Select this check box to enable custom sorting. Configure the sort order by selecting fields in the **Sort by** and **Sort by order** fields.
Sort by	Select any field in the list for sorting the spans in the timeline. Common practice is to select either the **Start date field** or the **End date field** as the sorting field. If you select a different sorting field, also include that field in the list of **Tooltip text fields** to give users a way of discovering the sort criteria.
Sort by order	Select the sort order for the sorting fields selected.
Interactive Options	
Allow horizontal moving?	Select this check box to permit users to drag timeline spans horizontally. Dragging changes the start and end dates and updates the record.
Allow start time dragging?	Select this check box to permit users to update the record by dragging the start time of a span.
Allow end time dragging?	Select this check box to permit users to update the record by dragging the end time of a span.
Range calculator	Specify a script include that calculates range restrictions and processes parent updates, if appropriate.

A completed Timeline Page record looks like this.



Name:	Release Timeline		
Table:	SCRUM release [rm_release_scrum]		
Start date field:	Planned start date		
End date field:	Planned end date		

Display Options

Show grid lines:	✓	Show span text:	✓
Show left pane:	☐	Span text fields:	Short description
Show summary pane:	✓	Show tooltips:	✓
Auto refresh:	Disabled	Tooltip text fields:	Number, Short description
CSS span color:	brown		

Filtering and Sorting

Condition: 5 records match condition

-- choose field -- | -- oper -- | -- value --

Perform custom sort:	✓
Sort by:	Planned start date
Sort by order:	Ascending

Interactive Options

Allow horizontal moving:	✓
Allow start time dragging:	✓
Allow end time dragging:	✓
Range calculator:	

Customizing the Timeline Page Span Styles

The **Timeline Page Span Style** related list allows you to define conditional span styles.

1. Navigate to **System UI > Timeline Pages**.
2. Open the timeline page for which you want to define the span style.
3. Go to the **Timeline Page Span Styles** related list and click **New**.
4. Fill in the form as described in the table and click **Submit**.

The Timeline Page Span Style form provides the following fields.

Field	Description
Condition	Create a condition to filter the results that appear in the timeline.
Label color	Select the color for the text under each span.
Label decoration	Select a character style for the text under each span: **Bold**, **Italic**, **Underline**, or **Line-through**.
Order	Enter a number to determine the sequence in which the style conditions are evaluated. Style conditions with a lower order are evaluated first.
Span color	Select the color of each span.
Timeline page	Select the timeline page to which the span style applies. By default, the span applies to the current timeline.

Timeline Sub Items

Use the **Timeline Sub Items** related list to define child spans for the timeline, based on records in a table that references the parent timeline's table. This can be used to generate a hierarchical relationship starting from a timeline page to any number of levels. For example, if there is a timeline page for a release, a sub item might be sprints, and a sprint might have stories as a sub item.

To create a new sub item:

1. Navigate to **System UI > Timeline Pages**.
2. Open the timeline page for which you want to add a sub item.
3. Go to the **Timeline Sub Items** related list and click **New**.
4. Fill in the the form as described in the table and click **Submit**.

The Timeline Sub Item form provides the following fields.

Field	Definition
Parent	[Read-only] Identifies the parent of the timeline sub item.
Name	Enter a unique name that describes the function of this timeline. For example, **Sprints for High Priority Changes**.
Table	Select the table called by this timeline. The selected table must have at least one reference field to the table selected for the parent timeline page. For example, if the parent timeline page uses the Scrum Release [rm_release_scrum] table, you might choose the Sprint [rm_sprint] table for a timeline sub item. **Note:** *The list shows only tables and database views that are in the same scope as the timeline (starting with the Fuji release).*
Start date field	Select a time-related field from the specified table to use as the start date for the timeline. For example, **Planned start date**.
End date field	Select a time-related field from the specified table to use as the end date for the timeline. For example, **Planned end date**.
Parent Reference Column	[Required] Select a reference field on which to base the timeline connection between the sub item records and the parent records. If multiple reference fields are available, choose the reference field that forms part of the hierarchy to be modeled by this timeline. If this list is blank, the sub item table contains no reference fields to the parent table. In this case, you must choose a different table for the sub item. ServiceNow uses the parent reference column to determine which records are displayed at each level of the timeline.
Display Options	
CSS span color	Enter a custom span color using any CSS color format, such as RGB or hexadecimal. If this field is blank, the default span color, light blue, is used.
Span text fields	Select fields from the specified table to have those values appear as span labels. For example, you might select **Number** and **Short description**. The span labels also appear in the left pane if the timeline displays the left pane.
Tooltip text fields	Select fields from the specified table to have those values appear as tooltips. For example, you might select **Category**, **Assigned to**, and **Due date** .
Filtering and Sorting	
Condition builder	Create a condition to filter the results that appear in the sub item. For example, you might create a condition that displays only active, high priority incidents.
Perform custom sort	Select this check box to enable custom sorting. Configure the sort order by selecting fields in the **Sort by** and **Sort by order** fields.
Sort by	Select any field in the list for sorting the spans in the timeline. Common practice is to select either the **Start date field** or the **End date field** as the sorting field. If you select a different sorting field, also include that field in the list of **Tooltip text fields** to give users a way of discovering the sort criteria.

Sort by order	Select the sort order for the sorting fields selected.
Interactive options	
Allow horizontal moving?	Select this check box to permit users to drag timeline spans horizontally. Dragging changes the start and end dates and updates the record.
Allow start time dragging?	Select this check box to permit users to update the record by dragging the start time of a span.
Allow end time dragging?	Select this check box to permit users to update the record by dragging the end time of a span.
Restriction	Specify the behavior when dragging a child span (available only if no **Range calculator** was specified for the parent timeline page). • **None:** No restriction is in place. • **Restrict by parent:** Child span can be moved only within the time frame defined by the parent span. • **Update parent:** Parent span is updated when the child span is moved outside the time frame defined by the parent span.

Making Timelines Visible to Selected Users

Make selected timelines available to users by creating a custom module within an application and defining the roles that can access it. To permit these users to update task records directly from the timeline, configure the timeline to allow span dragging.

Note: *Timelines delivered by a custom module are not entirely dynamic. The left pane, the summary pane, the auto-refresh feature, and the grid lines are* not *dynamic and do not reflect changes made to the timeline record after the module link is created. However, the data represented by the spans, the labels, and tooltips display all updates in the custom module.*

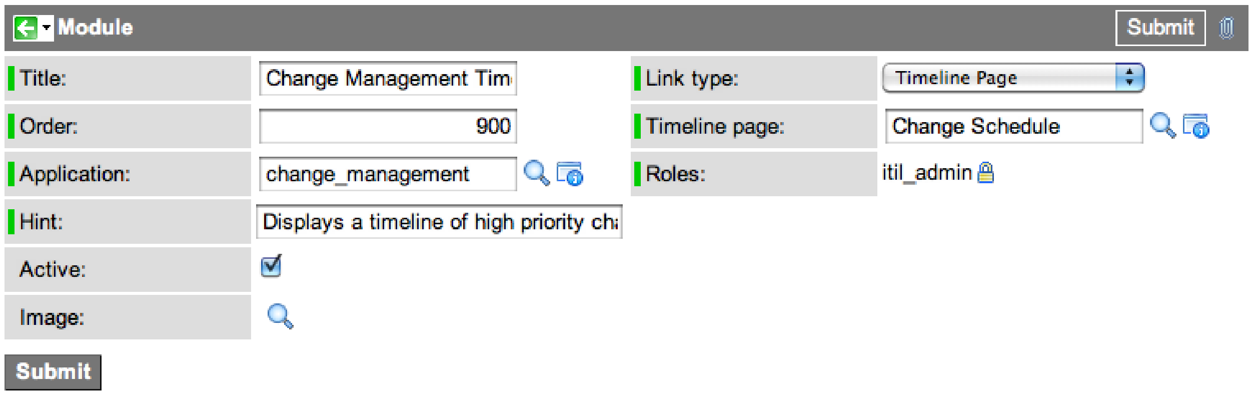
To create a timeline page module:

1. Right-click an application (such as Incident) in the navigation pane and select **Edit application**.
2. In the application **Modules** related list, click **New**.
3. Configure the Module form to add the **Timeline page** field.
4. Fill in the form as described in the table and click **Submit**.

The Module form provides the following fields.

Field	Description
Name	Enter the name of the module as it will appear in the navigation pane. For example, you might use **Planning Timeline**.
Order	Enter a number to determine the sequence in which this condition will be evaluated if more than one matching condition exists. Conditions with a lower order are evaluated first.
Application	Select the application for the new module.
Hint	Enter a brief description of the module that appears when the user places the cursor over the module name. For example, you might enter **Weekly view of high priority changes**.
Active	Select this check box to enable the module for the roles defined. Clear this check box to disable the module for all users.
Image	Select an appropriate icon to appear with the module name.
Link type	Select **Timeline Page**. When this link is selected, the **Timeline Page** field appears.
Timeline Page	Select the timeline page you want to appear in this module. For example, for the Change application select a change-related timeline, for the Incident application select an incident related timeline.
Roles	Select the roles that can access this module.

The completed module form looks like this.



Module

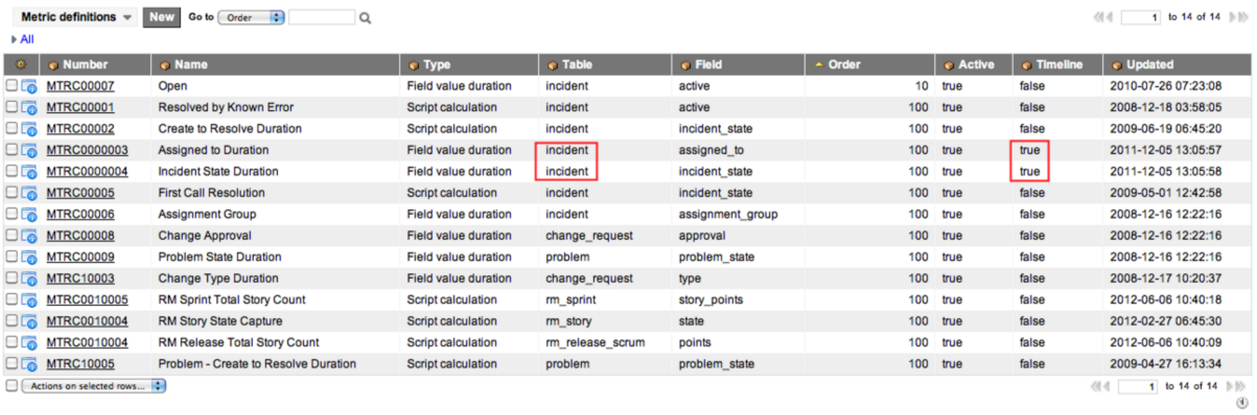
Title:	Change Management Tim	Link type:	Timeline Page
Order:	900	Timeline page:	Change Schedule
Application:	change_management	Roles:	itil_admin
Hint:	Displays a timeline of high priority ch		
Active:	✓		
Image:			

Displaying Metrics as Timelines

Administrators can allow users to display any metric on a timeline by activating the **Timeline Metrics** UI action.

1. Navigate to **Metrics > Definitions**.
2. Select the metric you want to display on a timeline. For example, Problem State Duration.
3. Select the **Timeline** checkbox.
4. Click **Update**.

The UI action is available on the same table as the metric. By default, only Incident metrics are available.



Metric definitions

Number	Name	Type	Table	Field	Order	Active	Timeline	Updated
MTRC00007	Open	Field value duration	incident	active	10	true	false	2010-07-26 07:23:08
MTRC00001	Resolved by Known Error	Script calculation	incident	active	100	true	false	2008-12-18 03:58:05
MTRC00002	Create to Resolve Duration	Script calculation	incident	incident_state	100	true	false	2009-06-19 06:45:20
MTRC0000003	Assigned to Duration	Field value duration	incident	assigned_to	100	true	true	2011-12-05 13:05:57
MTRC0000004	Incident State Duration	Field value duration	incident	incident_state	100	true	true	2011-12-05 13:05:58
MTRC00005	First Call Resolution	Script calculation	incident	incident_state	100	true	false	2009-05-01 12:42:58
MTRC00006	Assignment Group	Field value duration	incident	assignment_group	100	true	false	2008-12-16 12:22:16
MTRC00008	Change Approval	Field value duration	change_request	approval	100	true	false	2008-12-16 12:22:16
MTRC00009	Problem State Duration	Field value duration	problem	problem_state	100	true	false	2008-12-16 12:22:16
MTRC10003	Change Type Duration	Field value duration	change_request	type	100	true	false	2008-12-17 10:20:37
MTRC0010005	RM Sprint Total Story Count	Script calculation	rm_sprint	story_points	100	true	false	2012-06-06 10:40:18
MTRC0010004	RM Story State Capture	Script calculation	rm_story	state	100	true	false	2012-02-27 06:45:30
MTRC0010004	RM Release Total Story Count	Script calculation	rm_release_scrum	points	100	true	false	2012-06-06 10:40:09
MTRC10005	Problem - Create to Resolve Duration	Script calculation	problem	problem_state	100	true	false	2009-04-27 16:13:34

Viewing System Logs

Viewing System Logs

Note: *This article applies to Fuji and earlier releases. For more current information, see System Logs* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**'

Overview

Administrators can use various system logs to view information about system activity.

Logged Information

The following information is tracked in the system logs.

- Workflows
- Configuration
- Chats sessions
- Transactions for each view of each page in the system, including load times for network, server, and browser
- Inbound and outbound email
- Events triggered in the system
- Imports and integrations
- System warnings, errors, and script logs
- Upgrade information for any plugin activations, update sets, or system upgrades

Workflow Logging

The following workflow information is tracked in the system logs.

- Each activity executed, including:
  - Date and time started
  - Date and time ended
  - State, for example, Finished, Cancelled, Timed Out, Error
  - Result
  - Fault description, if there was an error
- Transition history, including:
  - Time of transition
  - Activity transitioned from
  - Activity transitioned to
  - Which transition was triggered
- Workflow log, including any log statements added to the workflow

Configuration Logging

The following configuration changes are tracked in the system logs.

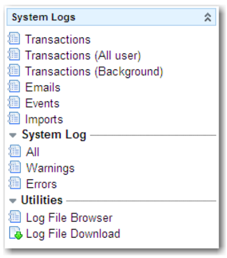
- Action taken, including insert, update, and delete
- Category of change
- Comments recorded with the change
- Name of the change
- XML difference of the change
- Update set the change is associated to
- Date and time of the change
- User who made the change
- Table where the change was made
- Name of the object being changed
- Type of object being changed
- View the change was made in, for changes to forms or lists

Log History

The system uses table rotation and table extension to archive older logs. By default, the system uses the following schedule to archive common logs:

Table	Archive Schedule	Rotations	Type
Event [ecc_event]	Every day	7	Rotation
Queue [ecc_queue]	Every day	7	Rotation
Event [sysevent]	Every day	7	Rotation
Log [syslog]	Every week	8	Rotation
Transaction Log [syslog_transaction]	Every week	8	Rotation
Email [sys_email]	Every 30 days	8	Extension

Available Logs



The following logs are available from the System Logs menu.

- **Transactions:** browser activities and background processes.
- **Email:**emails sent from within the platform.
- **Events:** events defined in the Event Registry.
- **Imports:** import activity within the platform.
- **System Logs** - warnings and errors triggered by the platform itself.

There are also log utilities provided:

- **Log File Browser:** browses the full log file.
- **Log File Download:** downloads the full log file.

Transactions

The transaction log records all browser activity for an instance of ServiceNow. This log provides the following information for all activities.

Field	Description
Created	Date and time of the browser action for the locale of the machine running the ServiceNow instance.
Created by	The user who created this activity.
Message	The user who created this activity. This field is no longer in use with the Calgary release; all user data comes from the **Created by** field.
Response time	Round trip response time for the browser request, in milliseconds
Network time	Latency time of the network response after the browser request is made, in milliseconds.
Output length	Size of the output string sent by ServiceNow to the browser, in bytes.
SQL count	Number of SQL server commands executed for this activity.
Business rule count	Number of ServiceNow business rules executed for this activity.
Business rule time	Elapsed time for the execution of the ServiceNow business rules for this activity.
URL	The application or module connected to by the client browser.
System ID	System generated identifier of the client instance making the request. This ID is used for cluster environments in which several instances (nodes) communicate with the database.
IP address	IP address of the client making the request.
GZipped	Indication of whether a compressed Web page was requested by the browser.
Protocol	The HTTP protocol used by the browser for this instance.

Slow Job Log

The **System Scheduler > Slow Job Log** module provides a transaction log filtered to show only slow transactions.

Client Transactions

For information on client transaction logging, see Client Transaction Timings.

Emails

The email log records all email notifications sent from all instances within the ServiceNow system. This is a verbose and unfiltered view of email. For a more detailed view, see the System Mailbox application.

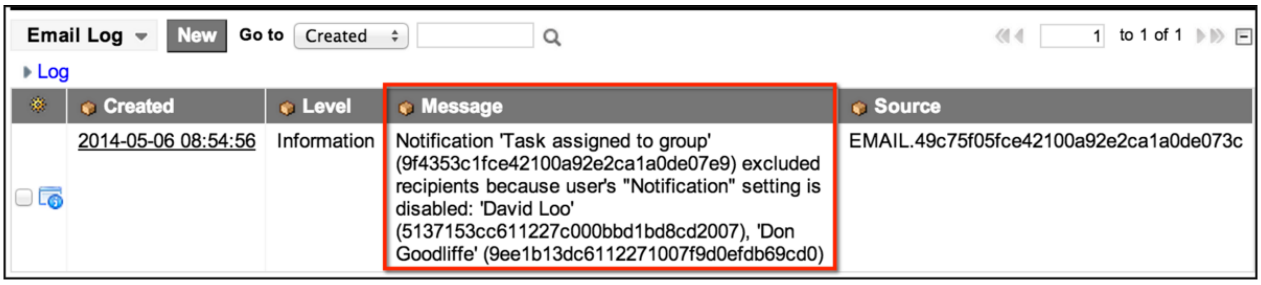
This log provides the following information for all notifications.

Field	Description
Mailbox	The system mailbox to use for filtering the email notifications displayed.
State	The current state of the notification (**Error**, **Ignored**, **Processed**, or **Ready**).
Receive type	The type of inbound email notification (**None**, **Forward**, **New**, or **Reply**).
Type	The status of the email notification. Choices are: • **received:** The server received this email. • **received - ignored:** The server received this email, but it was ignored by the instance for inbound email action purposes. Typically, these emails are either spam or auto-replies. See the **Error String** field for details. • **send - failed:** The server has attempted to send the email and failed. See the **Error String** field for details. • **send - ignored:** The server skipped sending this email. Typically, this is for an email which was generated but lacked a recipient email address or is a duplicate email. See the **Error String** field for details. • **send - ready:** The email is ready to be sent, but has not been sent out by the mail server. Typically, an email remains in this state for only a short time. • **sent:** The email was sent by the instance without any errors or issues.
Target	A Document ID reference to the record if the email is generated by an insert, update, or delete of a particular record.
User	The name of the user, from the user record, of the ServiceNow instance from which the email notification was sent. **Note:** This is a string field.
Notification Type	The type of notification. Choices are: • **None** • **SMS** • **SMTP**
UID	The unique ID for the server.
Created	The date and time of the email activity for the locale of the machine running the ServiceNow instance.
Deleted	An indication of whether the email was deleted from an instance mailbox.
Weight	The weight of the email, which determines the sending priority relative to other notifications on the same table. For more information, see Specifying When to Send the Notification.
Importance	An indication that the email was sent with a changed level of importance, such as **Urgent**.
Originating Event and Notification	An embedded list that stores the event and notification that initiated the email notification. For more information, see Events and Email Notification.
Subject	A configured description of the action that generated the email notification. You create the subject text for notifications in **System Policy > Email > Notifications**.
Error String	The error string captured from the email server to determine why the email was not sent. This is logged only if the email is **send-failed**.
Recipients	The email address of the recipient of each notification.
Body	The body of the email, displayed in raw HTML markup. Use the related link **Preview HTML Body** to see the body text as rendered HTML.
Content type	The email content type.
Headers	Any headers embedded in the email.

Outbound Email Notification Recipients

For outbound notifications, the email system log provides reasons that recipients were included or excluded. Each log entry corresponds to a reason for inclusion or exclusion. For example, all users who were excluded because they are inactive appear in a single log entry.

A series of system properties can be used to fine-tune the information to be logged. Two *master switch* properties, `glide.notification.recipient.include_logging` and `glide.notification.recipient.exclude_logging`, control all recipient inclusion and exclusion logging. Several other properties allow you to tailor the information reported in the logs to meet your needs. All of the properties are enabled by default.



Created	Level	Message	Source
2014-05-06 08:54:56	Information	Notification 'Task assigned to group' (9f4353c1fce42100a92e2ca1a0de07e9) excluded recipients because user's "Notification" setting is disabled: 'David Loo' (5137153cc611227c000bbd1bd8cd2007), 'Don Goodliffe' (9ee1b13dc6112271007f9d0efdb69cd0)	EMAIL.49c75f05fce42100a92e2ca1a0de073c

Events

The event log records all system events that occur within the ServiceNow system. This log provides the following information for all events that occur:

Field	Description
Created	Date and time of the event for the locale of the machine running the ServiceNow instance.
Name	Name of the event. You configure events in **System Definition > Business Rules**.
Parm1	Event-specific value that depends on the event and the recipient.
Parm2	Event-specific value that depends on the event and the recipient.
Table	Database table acted on for this event.
Processed	Date and time the event was processed This time reflects the locale of the machine running the ServiceNow instance.
Processing time	Time taken to process this event, in milliseconds.
Queue	Processor queue name.

Imports

The import log displays information in a verbose format about any data import activity within the ServiceNow platform. For a more detailed view of the import sets that produced a particular log, see **Import Sets > Transform History**.

This log provides the following information for all imports:

Field	Description
Created	Date and time of the import for the locale of the machine running the ServiceNow instance.
Level	Type of message displayed. For import files, the level is **Information**.
Message	System-generated message regarding the status of the import.
Source	Name of the external source of the import, such as an integration.

System Logs

System logs display warnings and errors within ServiceNow processes and records, and non-critical events such as memory usage on the ServiceNow server machine. This list view displays the log entries for the current day only. To view other log files, use the log file browser.

This log provides the following information for all occurrences:

Field	Description
Created	Date and time of the logging activity for the locale of the machine running the ServiceNow instance.
Level	Type of message displayed. The levels are **Debug**, **Error**, **Warning**, and **Information**. A warning is an error that has been handled and recovered. An error is something that must be fixed.
Message	System-generated message regarding the nature of the occurrence.
Source	Name of the process or area affected by the occurrence. For example, the source of the occurrence might be **EMAIL** or **Memory**.

Logging Utilities

ServiceNow provides the following logging utilities:

- Log file browser
- Log file download

Log File Browser

Use **System Logs > Log File Browser** to view any system log entry. You can search for log files by using the following filters:

Field	Description
Start time	Start date and time of the range you want to search, for the locale of the machine running the ServiceNow instance
Session ID	System-generated hexadecimal string that identifies the session that generated the log entry.
End time	End date and time of the range you want to search, for the locale of the machine running the ServiceNow instance
Message	System-generated description of the occurrence.
Level	Type of message displayed. The levels are **Debug**, **Error**, **Warning**, and **Information**. A warning is an error that has been handled and recovered. An error is something that must be fixed.
Thread name	System-generated identifier of the thread that created the log file.
Max rows	Maximum number of records returned for a particular filter.

Log File Download

ServiceNow creates compressed archives of system logs every two days and purges log archives after 21 days. You can download log file archives and view them with **System Logs > Log File Download**. Select a log archive from the list, and then click **Download log** under **Related Links** to open or save the archive.

System Diagnostics Application

The System Diagnostics application provides these logs that relate to the platform:

- **Upgrade History:** tracks every upgrade to an instance. See Upgrade History.
- **Slow Queries:** provides insight into how queries affect platform performance. See Slow Query Logs.

Customer Updates Table

Every change that is made in the system is recorded on the Customer Updates [sys_update_xml] table chronologically. To navigate to this table, enter **sys_update_xml.list** into the navigation filter.

The following information is stored about each update:

Field	Description
Name	A name that identifies the updated record.
Created	The date and time the customer update record was created.
Created By	The user who performed the change.
Type	The type of the update.
Updated	The date and time the customer update record was updated.
Updated By	The user who performed the update.
Updates	The number of times the record has been updated.
Target Name	The name of the element that was altered.
View	the view of the form that was altered if it was a form layout change.
Payload	The XML contents of the record after the change.
Remote Update Set	A reference to that update set if the change was performed by a remote update set.
Local Update Set	The update set the change is associated with.

For more information, see System Update Sets.

System Mailboxes

The **System Mailboxes** application contains the logs for all of the mail received or sent by the platform. For more information, see System Mailboxes.

Audited Tables

If a field on a particular table is audited, all changes to that field are tracked. For more information, see Turning on Auditing (History) for a Table.

This information is kept in two places:

- The History Sets table.
- The Audit table.

For a comparison of the two, see sys_audit versus sys_history_set.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/security/topic/p_SystemLogs.html

Client Transactions

Note: *This article applies to Fuji. For more current information, see Client Transaction Timings* [1] *at* http://docs.servicenow.com
The Wiki page is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

The Client Transaction Timings feature enhances the system logs by providing more information on the durations of transactions between the client and the server. By providing information on how time was spent during the transaction, performance issues can be tracked down to the source by seeing where the time is being consumed.

The system can collect information from the following browsers:

- Chrome
- Firefox
- Internet Explorer

Recording Client Transactions

Client transactions are stored with other transactions in the Transaction Log Entry [syslog_transaction] table. Client transactions all have the **client_transaction** field set to **true**.

Note: *Older data is managed on this table with the table cleaner.*

Enabling Client Transaction Logging

By default, the system does not gather client transaction data. To enable client transaction set the following system properties to **true**.

Property	Description
glide.client.track_transaction_timings	Enables the collection of client transactions.
glide.log.client.form.sections	Measures time rendering form sections.
glide.log.client.related.lists	Measures time rendering related lists.
glide.log.client.ui.policy	Measures time enforcing UI policies.
glide.log.client.script.on.load	Measures time running onLoad scripts.
glide.log.client.script.on.change	Measures time running onChange scripts.

Client Transactions Information

The **System Logs** application contains a **Client Transactions** module, which provides a list of every logged transaction between client and server within the last day. The following information is tracked:

Field	Description
Created	The moment the transaction was recorded.
Response Time	The number of ms spent by the server in fulfilling the transaction.
Business Rule Time	The number of ms spent by business rules triggered by the transaction.
SQL Time	The number of ms spent by the SQL database.
Client Response Time	(Load_completion_time) - (start_time). It is inclusive of server time.
Client Network Time	The number of ms spent by the network the client is connecting through.
Browser Time	The number of ms spent by the browser during the transaction.
Client Script Time	The number of ms spent executing client scripts.
UI Policy Time	The number of ms spent executing ui policy.
Type	Type of transaction (such as form or list). See Transaction Log Types.
Table	The table that was displayed. For example, incident, change_request.
View	The view for this form/list.

Transaction Log Types

The transaction log tracks the following transaction types:

Type	Description
List	List transactions.
Form	Form transactions.
XMLHttp	Transactions that run through GlideAjax, and the URL is xmlhttp.do.
Report	The page sys_report_template.do.
SOAP	SOAP transactions.
Export	When a list is exported as XML, Excel, and so forth.
Scheduler	When a scheduled job is performed.
Text Search	The text search transaction or any of its related operations.
Other	A generic type when no other type matches.

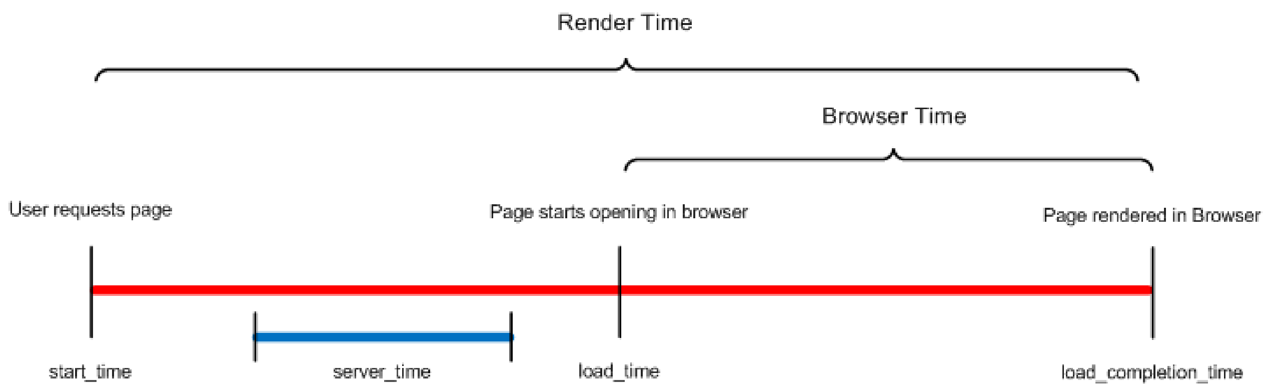
Client Detailed Information

A more detailed breakdown of the client timings for all Form rendering (but not list rendering) is also tracked. To see details, drill into a particular client transaction record and observe the related list at the base of the screen.

Field	Description
Order	The order during the load that this operation occurred
Type	The type of operation
Name	Descriptive name of this particular operation
Duration	Number of ms this particular operation took to complete

Timing Values

The following diagram illustrates the timing increments for rendering a page:



The variables in this diagram are defined as follows:

Variable	Description
start_time	The date and time the user requests a page (the user clicks on a link). This value is set by hooking into the **beforeunload** event of the previous page. The **beforeunload** event is not properly supported by WebKit browsers, which is why the client timings are not supported on Safari or Chrome.
load_time	The date and time that the current page starts loading in the browser. This value is set by an inline javascript that runs as the first script in the HTML body.
server_time	The time in ms spent by the server processing the transaction. The server reports this value to the client.
load_completion_time	The date and time that the page is fully rendered in the browser. This operation is performed as the last script on the page and identifies the time the page completed loading.

The **AJAXClientTiming** script include gathers and populates the following times in the Transaction Log Entry [syslog_transaction] table.

Label	Element	Description	Calculation
Response Time	client_response_time	Calculates the overall time to deliver the page by subtracting the time the user requests the page from the time the page is fully rendered in the browser.	load_completion_time - start_time
Network Time	client_network_time	Calculates the time the network takes to process the request by subtracting the time of the user's request from the time the page starts loading in the browser, and then subtracting the server processing time.	load_time - start_time - server_time
Browser Time	browser_time	Calculates the time the browser takes to deliver the page by subtracting the time the page is fully rendered from the time the page starts loading in the browser.	load_completion_time - load_time

Disabling Client Transactions

To disable client transaction functionality:

1. Enter **sys_properties.list** in the application navigator filter.
2. Locate the property named **glide.client.track_transaction_timings**.
3. Set the property value to **false**.

The functionality can be enabled again by setting the property value to **true**.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/reference/r_ClientTransactionTimings.html

Scheduling Events

Scheduling

Note: *This article applies to Fuji and earlier releases. For more current information, see Event Scheduling* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**'

Overview

There are a variety of tools available for scheduling actions or tasks to happen in the future.

Scheduled Jobs

Scheduled Jobs are scripts which can be set to be automatically performed at a specific date and time, or on a repeating basis. For more information, see Creating a Scheduled Job.

Event Registry

Events are registered in the **Event Log** by Business Rules. These events can be used to automate other activities, such as script actions or notifications. Events can be used to schedule actions or tasks to occur when conditions are fulfilled.

Examples:

- **kb.view** - an event triggered when a user views a Knowledge Base article, used to trigger the script action **Knowledge View** to create a **Knowledge Use** record every time an article is viewed.
- **incident.commented** - an event triggered when a user comments on an article, used by two **incident commented** email notifications.

For more information, see Event Registry.

Inactivity Monitors

An inactivity monitor triggers an event if a record has not been updated for a defined length of time. For more information, see Setting up Inactivity Monitors.

Script Actions

Script actions are scripts which are triggered when an event is recorded in the log. In that way, scripts can be set to be performed whenever a particular activity occurs in the platform, rather than at a particular time (like Scheduled Jobs) or in response to particular conditions (like a Business Rule).

Notifications

Events are also used to trigger Email Notifications when an event is recorded in the log. For more information, see Email Notifications.

Scheduled Reports

Once reports are defined, they can be scheduled to be emailed at a specific time, or at regular intervals, using the reporting interface. For more information, see Scheduling Reports.

On-Call Rotation

The Group On-Call Rotation Plugin allows a schedule to be defined to determine what users are primary contacts during particular hours of the day. For more information, see Group On-Call Rotation Plugin.

Scheduled Workflows

Workflows provide a robust system for automating advanced multi-step processes. Workflows can be triggered by conditions, like Business Rules, or they can be scheduled for a particular time/recurring schedule, like Scheduled Jobs. For information, see Scheduling a Workflow

Maintenance Schedules

Changes to the CMDB can be managed through the Maintenance Schedules Plugin, which allows changes to be proposed and viewed through a timeline. For more information, see Maintenance Schedules Plugin.

References

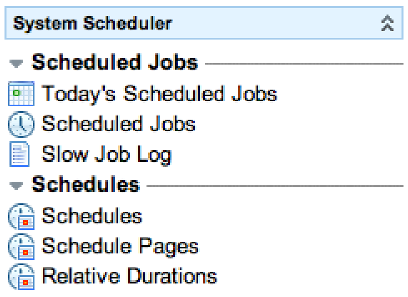
[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/concept/c_ScheduleEvents.html

System Scheduler

Note: *This article applies to Fuji and earlier releases. For more current information, see System Scheduler* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.'**

Overview

The System Scheduler application contains two separate engines for scheduling: the Scheduled Jobs engine and the Schedules engine. Functionality described here requires the Admin role.



Scheduled Jobs

Scheduled Jobs performs any work that must be done at a specific time or on a recurring basis. The Scheduled Jobs module links to the Schedule [sys_trigger] table. Manipulating records on the Schedule table is not recommended. Use this table to view existing base system scheduled jobs.

The Scheduled Jobs module in System Definition is an admin-friendly front end for scheduling work. Use this module to create new scheduled jobs.

For more information, see Creating a Scheduled Job.

Schedules

Schedules are rules which include or exclude time on a calendar. They are used by service levels, inactivity monitor, and group on-call rotation. For instance, a schedule can be defined to restrict service levels to only apply to weekdays during business hours, or to exclude holidays from an on-call rotation.

For more information, see Using Schedules.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/concept/c_SystemScheduler.html

Creating a Scheduled Job

Note: *This article applies to Fuji. For more current information, see Scheduled Jobs* [1] *at* http://docs.servicenow.com The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

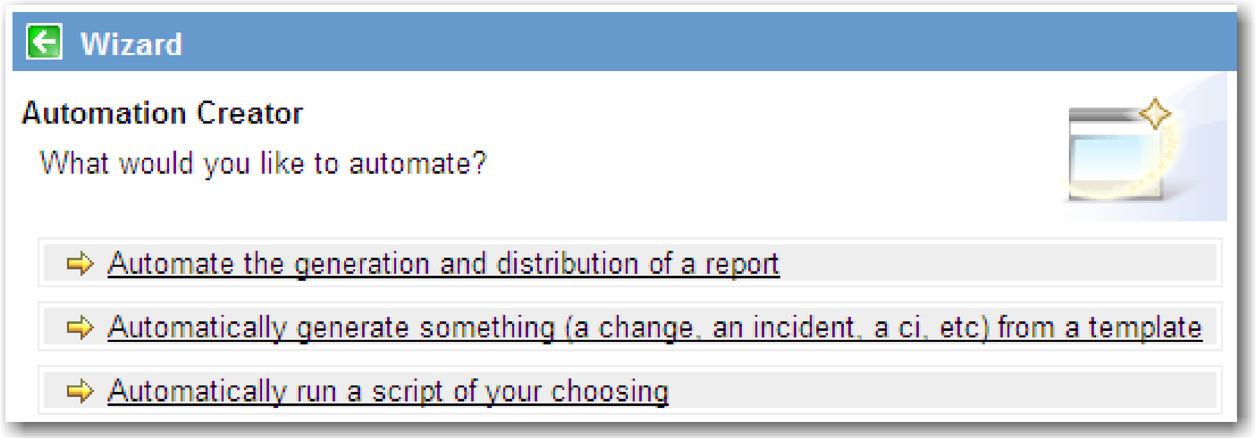
Scheduled Jobs are automated pieces of work that can be performed at either a particular time, or on a recurring schedule. These kinds of tasks can be automated:

- Run and distribute a report
- Generate a record (incident, change, configuration item, etc.) from a template
- Run a business rule and do whatever the rule contains

Note: The method described below creates a scheduled job on the Schedule Job [sysauto] table (**System Definition > Scheduled Jobs**). Create all new scheduled jobs using this method. Some existing scheduled jobs are found on the Schedule Item [sys_trigger] table (**System Scheduler > Scheduled Jobs**). Do *not* create new scheduled jobs on the Schedule Item table.

Creating a Scheduled Job

To create a scheduled job, navigate to **System Definition > Scheduled Jobs** and click **New**. Select the appropriate type of scheduled job.


Wizard
Automation Creator
What would you like to automate?
Automate the generation and distribution of a report
Automatically generate something (a change, an incident, a ci, etc) from a template
Automatically run a script of your choosing

The fields presented will depend on the type of scheduled job required.

Scheduling a Report

Scheduled Reports are distributed via email. Scheduled reports created by an individual whose user account is deactivated might not display any data. To ensure that the desired data is displayed, an active user must recreate the scheduled report.

Populate the following fields to schedule a report:

Field	Input Value
Name	A name to identify this scheduled job.
Report	A reference to the report to be scheduled. Calendar reports and Pivot Table reports are not currently supported for automatic emailing. For information on creating or modifying reports, see Generating Reports.
Users	The users to send the report to at the scheduled date and time. The user must have Notification set to "Email" on their user record to receive reports. To force users to receive emails, use the **Email Addresses** field.
Groups	The groups to send the report to at the scheduled date and time.
Email addresses	Any other email addresses to send the report to, separated by commas. These emails will always receive the report, even if there's a matching user record for that address that says "Do not notify".
Active	If true, this report will be sent at the scheduled date and time.
Run	The type of schedule to send the report on. Choices are: • Daily • Weekly • Monthly • Periodically • Once
Day	• If **Run** is **Weekly**, the day of the week. • If **Run** is **Monthly**, the day of the month.
Repeat Interval	If **Run** is **Periodically**, the duration between each scheduled report. The duration can be days, hours, or minutes.
Time	If **Run** is **Weekly** or **Monthly** the time of day, on a 24 hour clock.
Starting	The date and time of the first scheduled report.
Priority	The numerical priority of a scheduled job. Set essential jobs to a priority value below **100** and nonessential jobs to a priority above **100**. If 70% or more of all scheduled jobs are **Overdue**, any jobs marked with a value above 100 do not run.
Subject	The subject line for the email.
Introductory Message	The body of the email.
Run as	The user creating and running the scheduled job. To have the scheduled job assigned to system behavior instead of the person creating the scheduled job, create a system or “dummy” user and add that user to this field.
Type	The file-type of the attached report. Choices are: • PDF-landscape • PDF • Excel • CSV • PNG
Include Detail	If checked, will include details on the records in the report.
Zip Output	If checked, the report will be compressed in a .zip file.

As a workaround for scheduling calendar and pivot table reports for email distribution, consider using the publish option. Publish creates a URL for the report and displays the address above the report form. You can create an email notification with this URL and send the link to people who need to see the report, or you can send the url for the calendar report to a distribution list. For details about publishing reports, see Publishing Reports for more information.

Scheduling Entity Generation

To schedule the generation of an entity, populate the following fields:

Field	Input Value
Name	A name to identify this scheduled entity generation.
Active	If true, the entity will be generated at the scheduled date and time.
Run	The type of schedule to generate the entity on. Choices are: • Daily • Weekly • Monthly • Periodically • Once
Day	• If **Run** is **Weekly**, the day of the week. • If **Run** is **Monthly**, the day of the month.
Repeat Interval	If **Run** is **Periodically**, the duration between each scheduled generation.
Time	If **Run** is **Weekly** or **Monthly** the time of day, on a 24 hour clock.
Starting	The date and time of the first scheduled generation.
Conditional	If checked, the entity will only be generated if certain conditions are met.
Condition	If **Conditional** is checked, a script determines under what conditions the scheduled script is executed. The last expression of the script should evaluate to a Boolean (true/false) value. See Example Conditional Script.
Generate this	A reference to a template for a record.

Scheduling Script Execution

Field	Input Value
Name	A name to identify this scheduled script execution.
Active	If true, the script will be executed at the scheduled date and time.
Run	The type of schedule to execute the script on. Choices are: • Daily • Weekly • Monthly • Periodically • Once
Day	• If **Run** is **Weekly**, the day of the week. • If **Run** is **Monthly**, the day of the month.
Repeat Interval	If **Run** is **Periodically**, the duration between each script execution.
Time	If **Run** is **Weekly** or **Monthly** the time of day, on a 24 hour clock.
Starting	The date and time of the first scheduled script.
Priority	The numerical priority of a scheduled job. Consider setting async business rules to a priority of **100** or above to ensure any data import scheduled jobs have time to run first.

Application	The application that contains the script (available starting with the Fuji release).
Conditional	If checked, the script will only be executed if the **Conditions** evaluate to true.
Condition	If **Conditional** is checked, a script determines under what conditions the scheduled script is executed. The last expression of the script should evaluate to a Boolean (true/false) value. See Example Conditional Script.
Run this script	The script to run at the scheduled date and time.

Example Conditional Script

The following is an example of a conditional script. This example runs the scheduled job only if there are active Incidents older than 30 days.

```
// Only run this Scheduled Job if there are active Incidents over 30
days old
var ga = new GlideAggregate('incident');
ga.addAggregate('COUNT');
ga.addQuery('active', 'true');
ga.addQuery('sys_created_on', '<', gs.daysAgo(30));
ga.query();
ga.next();
ga.getAggregate('COUNT') !== '0'
```

Scheduling Special Cases

Scheduling for the End of the Month

Because months have different lengths, take care when scheduling jobs for the end of the month.

- Scheduling an event for the 29th or 30th is not recommended, because the scheduled job will not be executed in months (like February) which are shorter than those dates.
- If an event is scheduled for the 31st, it will be executed on the last day of the month, even if the month is shorter. For example, something scheduled to run on the 31st of the month will run on February 28th or February 29th in a Leap Year.

Scheduling for Weekdays

For scheduled scripts, use the following script to run only on weekdays:

```
function checkWeekdays() {
      var now = new Date();
      var day = now.getDay();
      var result = false;
      if(day != 0 && day != 6) {
            result = true;
      }
      return result;
}
checkWeekdays();
```

Executing Scheduled Jobs from Scripts

To execute a scheduled job triggered by an event, use the following script:

Note: *This API call changed in the Calgary release:*

- *SncTriggerSynchronizer* replaces *Packages.com.snc.automation.TriggerSynchronizer*

The new script object calls apply to the Calgary release and beyond. For releases prior to Calgary, substitute the packages call as described above. Packages calls are not valid beginning with the Calgary release. For more information, see Scripting API Changes.

```
//Execute a scheduled script job
var rec = new GlideRecord('sysauto_script');
rec.get('name', 'YOUR_JOB_NAME_HERE');
SncTriggerSynchronizer.executeNow(rec);
```

This script can be run using one of several tables:

- **scheduled_import_set** (Scheduled Import Sets)
- **sysauto_script** (Scheduled Script Execution)
- **sysauto_template** (Scheduled Template Generation)
- **sysauto_report** (Scheduled Report)

Note that `SncTriggerSynchronizer` does not provide methods to execute scheduled jobs in the future.

Running Scheduled Jobs Imported from Another Instance

To prevent unexpected data changes, the system does not automatically create Schedule Item [sys_trigger] records for Scheduled Jobs [sysauto] imported from an XML file such as an update set. To run a scheduled job imported from another instance, update the scheduled job record.

Viewing Schedule Items

Schedule items are individual instances of a scheduled job. To see which scheduled jobs will run today, navigate to **System Scheduler > Today's Scheduled Jobs**. The table displays each schedule item that will be run. It is usually inadvisable to modify the schedule items themselves. It is best to modify the scheduled jobs themselves.

References

[1] https://docs.servicenow.com/bundle/istanbul-servicenow-platform/page/administer/reference-pages/concept/c_ScheduledJobs.html

Setting Inactivity Monitors

Note: *This article applies to Fuji. For more current information, see Set an Inactivity Monitor* [1] *at* http://docs.servicenow.com
The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

An inactivity monitor triggers an event for a task record if the task has been inactive for a certain amount of time. If the task remains inactive, the monitor repeats at regular intervals. User updates to the task record restart the monitor. If **Reset Conditions** are defined for the monitor but have not been met when you update the task record, the monitor is not restarted. Inactivity monitors only apply to records on tables that extend the Task table, or the Task table itself.

When an inactivity monitor triggers, it generates an event in the form **<tablename>.inactivity** (for example, **incident.inactivity**). The inactivity monitor does not automatically specify further actions, so you must either define either an email notification or script action to drive further action.

Activity

A record's activity is only based on user updates. System updates do not count as activity.

Setting up an Inactivity Monitor

1. Navigate to **System Policy > Inactivity Monitors** and click **New**.
2. Give the inactivity monitor a name.
3. Specify the type of record to monitor in the **Table** field.
4. Specify how long the inactivity monitor should wait before sending each notification in the **Wait** field.
5. Specify any additional conditions in the **Condition** field. At least one condition must be specified for the inactivity monitor to work.
6. Specify an **Order** if multiple inactivity monitors might have their conditions met for a given record - the one with the lowest order will be used.
7. Click **Save**.

Note: *If conditions are changed on an inactivity monitor, the monitor stops tracking previously tracked records. An inactivity monitor does not track records that were created before the inactivity monitor, even if the record meets all other conditions.*

Escalation Intervals and Pause Conditions

Escalation Intervals and Pause Conditions are not relevant to an inactivity monitor. The related list and field are available because the inactivity monitor table extends the table used for SLAs, but these elements are not used in any way when an inactivity monitor attaches or is triggered.

Enhancements

Calgary

The following enhancements are available with the Calgary release.

- The **Wait** field on the SLA inactivity monitor form is now mandatory.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/task/t_SetAnInactivityMonitor.html

Event Registry

Note: *This article applies to Fuji and earlier releases. For more current information, see Event Registry* [1] *at* http://docs.servicenow.com **The ServiceNow Wiki is no longer being updated. Visit** http://docs.servicenow.com **for the latest product documentation.**'

Overview

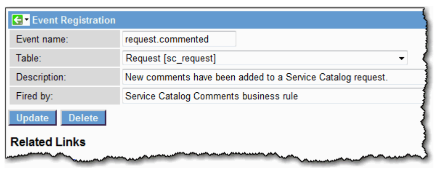
After you create a new event and a business rule that uses the event, you must register it. Registration lets other parts of the system, such as Email Notifications and Script Actions, see the event in their list of available events and react to the event when it occurs.

Registering an Event

To register an event, browse to **System Policy > Events > Registry**, and then click **New**.

Complete the Event Registration form as follows:

Field	Description
Name	The name of your new event.
Table	The database table for this event. **Note:** *The list shows only tables and database views that are in the same scope as the event (starting with the Fuji release).*
Description	Short description of the purpose of the event.
Fired by	Name of the business rule that runs the event. This field is for reference only and is not used by any process. Make sure there is enough information to locate your event again.
Queue	Name of the queue that the event is placed into when triggered.


Event Registration

Event name:	request.commented
Table:	Request [sc_request]
Description:	New comments have been added to a Service Catalog request.
Fired by:	Service Catalog Comments business rule

Sample event registry form

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/platform-events/concept/c_EventRegistry.html

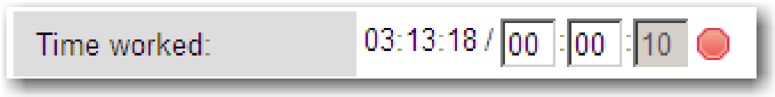
Timing

Note: *This article applies to Fuji. For more current information, see Timing Functionality* [1] *at* http://docs.servicenow.com The ServiceNow Wiki is no longer being updated. Please refer to http://docs.servicenow.com for the latest product documentation.

Overview

Timing functionality are tools that exist to answer the question "How Long?"

Time Worked Fields


Time worked: 03:13:18 / 00 : 00 : 10

The Task [task] table provides a time-tracking field called **Time worked**. This field measures how long a record has been viewed in order to measure work time on a ticket. Any table that extends Task can use this field. To add the field, simply personalize the form.

As the record is viewed, the timer counts upward. To pause the timer, click the stop icon (🔴); to resume the timer, click the start icon (▶).

When the task is saved, the amount of new time in the timer is used to generate a record on the Time Worked [task_time_worked] table. This table can be viewed as a related list on the task form.

By default, the time displayed in the **Time worked** field displays a cumulative value stored in the task record. If you modify a Time Worked record, the changes will not be reflected in the task timer.

You can set the property `com.snc.time_worked.update_task_timer` to enable updating of the task timer value based on changes to the time worked records. This is accomplished through the **Update task timer** business rule.

SLAs

Service Level Agreements time how long a task meets a certain condition, and is primarily used to ensure that tasks are handled within a pre-determined time limit.

SLAs define the following conditions:

- Start Conditions
- Pause Conditions
- Stop Conditions

Once a task meets the Start Conditions, the SLA will time how long the task remains in that condition (unless it meets Pause Conditions). The timer will end if the Stop Conditions are met. If the time-limit is passed, the SLA will be marked **breached**.

Notifications can be driven off of the SLA to warn interested parties as the time limit approaches.

Metric Definitions

Defined metrics can track how long an audited field holds a certain value. For instance, a metric can track how long an incident is assigned to an individual, or how long an incident is in the state **Active**. For more information on using metrics to time, Metric Definition Plugin.

References

[1] https://docs.servicenow.com/bundle/jakarta-servicenow-platform/page/administer/time/concept/c_TimingFunctionality.html

Calculating Time

Relative Durations

Overview

Relative Duration allows an event to be triggered at a time relative to the current time which might vary based on what the current time is. These relative durations are used out-of-box in SLAs and Workflow Activities.

Examples

The following examples are commonly used **Relative Durations**:

- **End of next business day** - sets the duration to the amount of time between the current time and 5:00PM at the end of the next business day:
  - If the current day is Monday-Thursday, it would be set to the end of the next day.
  - If the current day is Friday-Sunday., it would be set to the end of Monday
- **2 business days by 4PM if before 10AM** - sets the duration to the amount of time between the current time and two business days from now if before 10 AM.
  - If the event is generated before 10:00AM, it would be set to 4PM two business days later.
  - If the event is generated after 10:00AM, it would be set to 4PM three business days later, ignoring the first day.

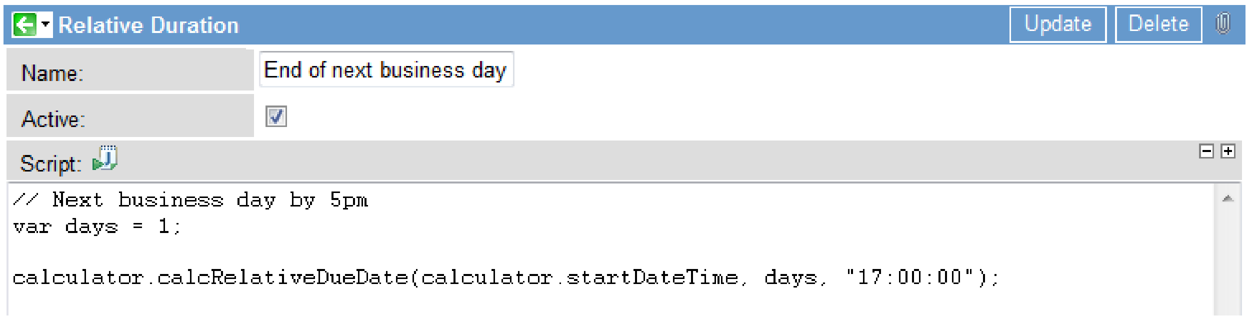
Note: *Business days have the same 24-hour duration as calendar days unless a schedule is defined.*

Defining Relative Durations

From the left navigation pane, select **System Scheduler > Relative Durations** to create or edit **Relative Durations**. Out of the box there are four Relative Durations in the system:

- 2 business days by 4pm
- 3 business days by 4pm
- Next business day by 4pm
- End of next business day.

We will start by looking at the "End of next business day" Relative Duration. From the Relative Durations list, "End of next business day" Relative Duration. The variable "days" is set to one, because we want the result of this calculation to land one day in the future. The rest of the script is as in the screenshot below. If desired, you can customize the time at which you want the Relative Duration to end (currently set to 5pm).



Name: End of next business day

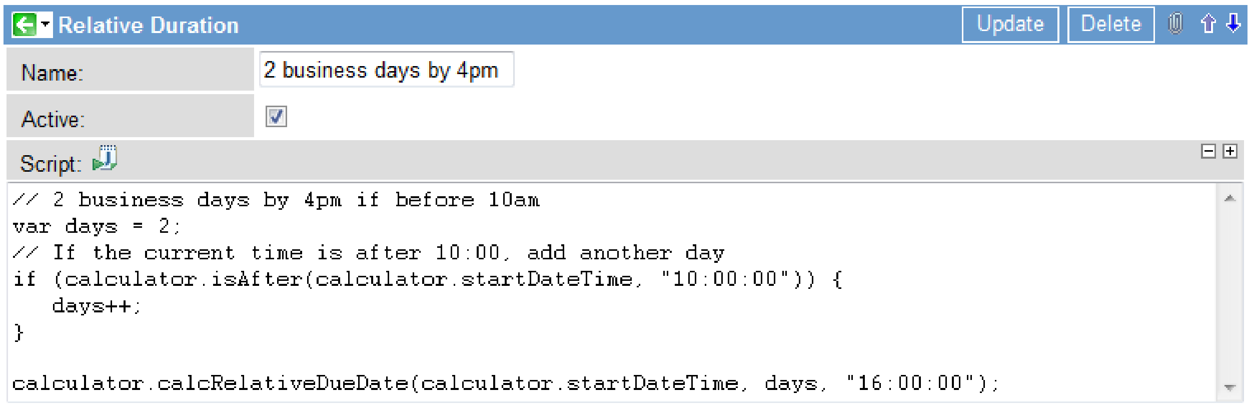
Active: ☑

Script:

```
// Next business day by 5pm
var days = 1;

calculator.calcRelativeDueDate(calculator.startDateTime, days, "17:00:00");
```

There is one more important Relative Duration design aspect that is used by the other three out-of-box Relative Durations. To illustrate this design we will look at "2 business days by 4pm".

As you can see in the image below, within the script there is an if-statement. This if-statement is checking to see if the calculated time is after 10am. If it is, then an additional day is added to the calculation. Hence the description of "2 business days by 4pm if before 10am"



Name: 2 business days by 4pm

Active: ☑

Script:

```
// 2 business days by 4pm if before 10am
var days = 2;
// If the current time is after 10:00, add another day
if (calculator.isAfter(calculator.startDateTime, "10:00:00")) {
   days++;
}

calculator.calcRelativeDueDate(calculator.startDateTime, days, "16:00:00");
```

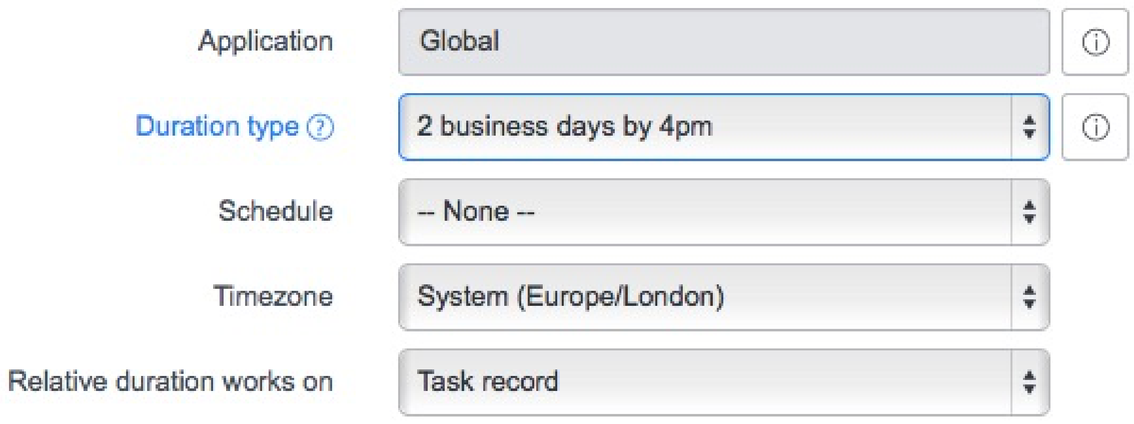
Note that "End of the business day" has nothing to do with the associated Schedule. The end time of 17:00 is hardcoded into this Relative Duration script. If you want the time to be different than the out of the box 5pm, you must change it in the script.

Using Relative Durations with SLAs

Relative Durations can be called from SLAs.

Selecting the Record to Calculate Relative Durations Against

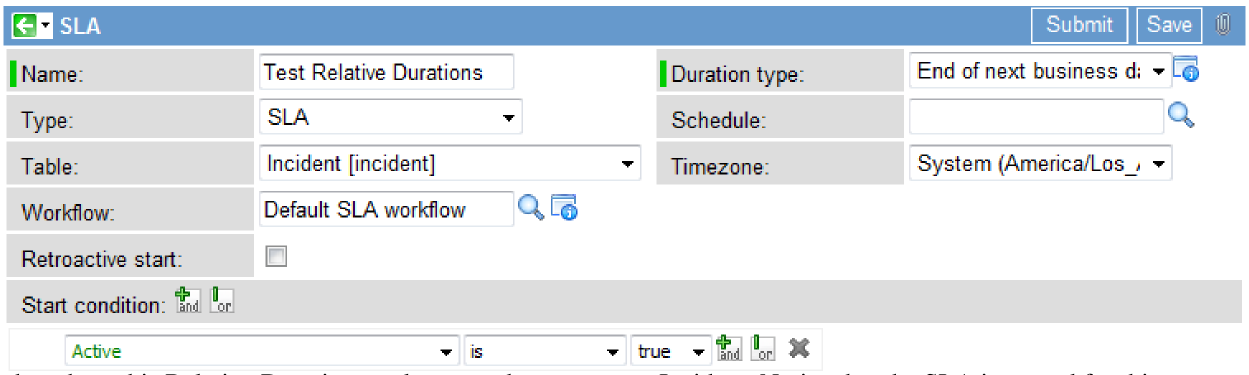
When defining an SLA, you can also define the record against which the relative duration should be calculated, using the **Relative duration works on** field to select to calculate either against the SLA record or the task record that the SLA record is used for.



Application	Global
Duration type	2 business days by 4pm
Schedule	-- None --
Timezone	System (Europe/London)
Relative duration works on	Task record

Example: Using Relative Durations with SLAs

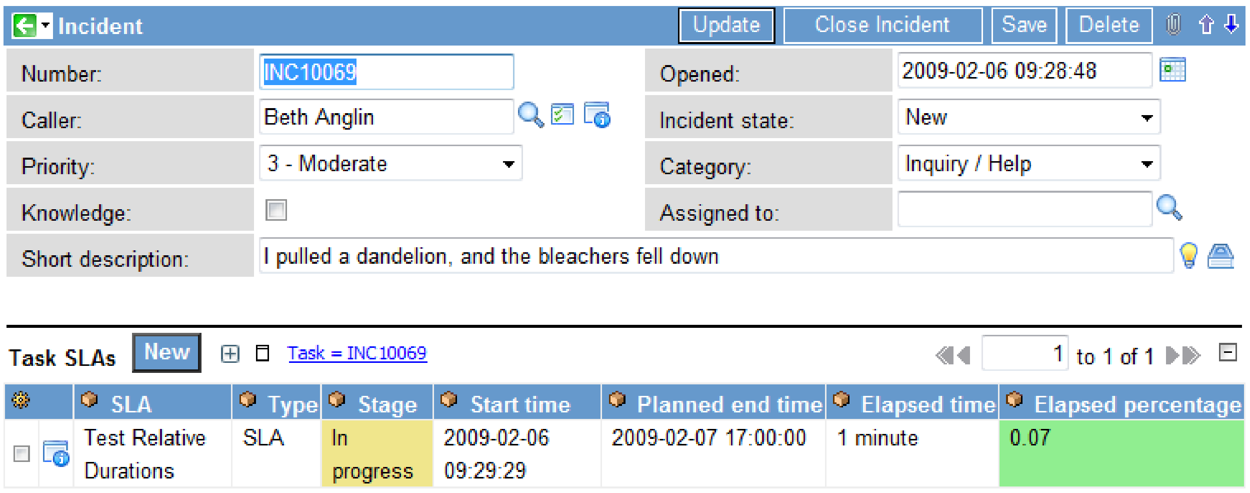
1. Create an SLA that has a Relative Duration of "End of next business day".
2. Complete the rest of the fields of this SLA with the values as shown below, also setting a Schedule and a Timezone if you want.



Name:	Test Relative Durations	Duration type:	End of next business d
Type:	SLA	Schedule:	
Table:	Incident [incident]	Timezone:	System (America/Los_
Workflow:	Default SLA workflow		
Retroactive start:	☐		
Start condition:			

Active is true

3. To show how this Relative Duration works, go and create a new Incident. Notice that the SLA is started for this incident. If you look at the "Planned End Date" field you will notice that the date is the next business day at 5pm:



Incident

Number:	INC10069	Opened:	2009-02-06 09:28:48
Caller:	Beth Anglin	Incident state:	New
Priority:	3 - Moderate	Category:	Inquiry / Help
Knowledge:	☐	Assigned to:	
Short description:	I pulled a dandelion, and the bleachers fell down		

Task SLAs — Task = INC10069 — 1 to 1 of 1

SLA	Type	Stage	Start time	Planned end time	Elapsed time	Elapsed percentage
Test Relative Durations	SLA	In progress	2009-02-06 09:29:29	2009-02-07 17:00:00	1 minute	0.07

Note: *Pause conditions are not compatible with Relative Durations.*

Article Sources and Contributors

Introduction to Time-Related Functionality *Source*: http://wiki.servicenow.com/index.php?oldid=130248 *Contributors*: Cheryl.dolan, Emily.partridge, G.yedwab, Joe.Westrich, Joseph.messerschmidt

Using Date/Time Fields *Source*: http://wiki.servicenow.com/index.php?oldid=250398 *Contributors*: Amy.bowman, Annmarie, Emily.partridge, Fuji.publishing.user, G.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Neola, Phillip.salzman, Rachel.sienko

Using Time Zones *Source*: http://wiki.servicenow.com/index.php?oldid=250408 *Contributors*: CapaJC, Cheryl.dolan, Chris.maloy, Emily.partridge, Fuji.publishing.user, G.yedwab, Jim.uebbing, Joe.Westrich, John.ramos, Joseph.messerschmidt, Phillip.salzman, Publishing.user, Rachel.sienko, Steven.wood, Suzanne.smith

Using Schedules *Source*: http://wiki.servicenow.com/index.php?oldid=251012 *Contributors*: Cheryl.dolan, Fuji.publishing.user, G.yedwab, Guy.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Phillip.salzman, Publishing.user, Rachel.sienko, Suzanne.smith, Vaughn.romero

Displaying Time *Source*: http://wiki.servicenow.com/index.php?oldid=250540 *Contributors*: G.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt

Schedule Pages *Source*: http://wiki.servicenow.com/index.php?oldid=250837 *Contributors*: Emily.partridge, Eric.jacobson, G.yedwab, Gadi.yedwab, George.rawlins, Guy.yedwab, Jim.uebbing, Joe.Westrich, John.ramos, Joseph.messerschmidt, Mark.johnson, Neola, Rachel.sienko, Steven.wood

Timeline Pages *Source*: http://wiki.servicenow.com/index.php?oldid=241874 *Contributors*: Dawn.bunting, Emily.partridge, Fuji.publishing.user, G.yedwab, Guy.yedwab, Joe.Westrich, Joseph.messerschmidt, Mark.johnson, Steven.wood, Vaughn.romero, Wallymarx

Viewing System Logs *Source*: http://wiki.servicenow.com/index.php?oldid=251048 *Contributors*: Emily.partridge, G.yedwab, Guy.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Neola, Peter.smith, Publishing.user, Rachel.sienko, Vaughn.romero

Client Transactions *Source*: http://wiki.servicenow.com/index.php?oldid=82837 *Contributors*: Cheryl.dolan, Emily.partridge, G.yedwab, Guy.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Pat.Casey, Peter.smith, Phillip.salzman, Publishing.user, Rachel.sienko, Steven.wood, Vaughn.romero, Wallymarx

Scheduling *Source*: http://wiki.servicenow.com/index.php?oldid=250841 *Contributors*: G.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt

System Scheduler *Source*: http://wiki.servicenow.com/index.php?oldid=250932 *Contributors*: CapaJC, Emily.partridge, G.yedwab, Guy.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Steven.wood, Vhearne

Creating a Scheduled Job *Source*: http://wiki.servicenow.com/index.php?oldid=250261 *Contributors*: Emily.partridge, Fuji.publishing.user, G.yedwab, George.rawlins, Guy.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Michael.randall, Rachel.sienko, Steven.wood, Suzanne.smith, Vaughn.romero

Setting Inactivity Monitors *Source*: http://wiki.servicenow.com/index.php?oldid=250368 *Contributors*: CapaJC, Emily.partridge, G.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Steven.wood, Vaughn.romero

Event Registry *Source*: http://wiki.servicenow.com/index.php?oldid=250568 *Contributors*: Fuji.publishing.user, G.yedwab, Jim.uebbing, Joe.Westrich, John.ramos, Joseph.messerschmidt, Steven.wood

Timing *Source*: http://wiki.servicenow.com/index.php?oldid=250383 *Contributors*: G.yedwab, Joe.Westrich, John.ramos, Joseph.messerschmidt, Sydney.nickell

Relative Durations *Source*: http://wiki.servicenow.com/index.php?oldid=246675 *Contributors*: David.Bailey, G.yedwab, Guy.yedwab, Joe.Westrich, Joseph.messerschmidt, Suzanne.smith

Image Sources, Licenses and Contributors