ACL
Cancer Research UK

Advanced Computation Laboratory,
Cancer Research UK
44 Lincoln's Inn Fields
London WC2A 3PX
Tel: 020 7269 3627 Fax: 020 7269 3186

CREDO-2002-005

# Tallis 1.4 Help

Ayelet Oettinger & Tony Rose

| | |
|---|---|
| **Creation date** | 06 September 2005 |
| **Last modification** | 06 September 2005 |
| **Last edited by** | Ayelet Oettinger |
| **Revision** | 1 |
| **Version** | 1 |
| **Circulation** | Unrestricted |

# 1. Introduction

## 1.1 What can I do with Tallis

Tallis Composer can be used to create and edit *Process Descriptions*, i.e. descriptions, in the PRO*forma* language, of the tasks that are to be carried out by interacting agents (human beings or software components) in order to accomplish some objective.

Process Descriptions can be enacted by the *Tallis Engine*. The engine keeps track of which tasks need to be performed to advance the process, and provides information to external agents regarding the current state of the process. The engine can also receive messages from agents indicating that they have completed certain tasks or provided data relevant to the running of the process.

## 1.2 What's new in Tallis 1.4?

The major difference between Tallis 1.4 and Tallis 1.3 is in the User Interface – this has been significantly redesigned and (we hope) greatly improved following feedback from both users and usability professionals. Our work in this respect is by no means complete, but we have nonetheless started a process that we hope will ultimately lead to a much more professional and usable system.

Tallis 1.4 also contains a number of new features and capabilities. Notable among these are:

- Access to an extended online process repository, with sophisticated user/group/folder permissioning
- Improved support for process indexing & retrieval through the use of descriptive metadata (based on Dublin Core)
- Access to extensive online help

In addition, Tallis 1.4 contains a number of bug fixes. In particular, many of the bugs relating to the layout of the various dialogue boxes in the Tallis Tester (notably the Data Entry dialogue) have now been fixed. We have also addressed some of the major Tallis Engine bugs that had been logged on Bugzilla (see Note).

**Note**

Please log any future bug reports on http://acl.icnet.uk/bugs/.

## 1.3 Application Support

For general enquiries, comments or suggestions about the use of Tallis, please refer to the Tallis User Group: http://groups.yahoo.com/group/tallisusers/

To report bugs, please refer to Bugzilla: http://acl.icnet.uk/bugs/

For any other enquiries, please email tallis_support@acl.icnet.uk

# 2. Installation

## 2.1 Installing and updating Tallis

The Tallis Authoring Suite consists of two applications: The Tallis Composer (for authoring process descriptions) and the Tallis Tester (for enacting/testing process descriptions). Both of these are bundled as part of the standard download, which is available in two forms:

1. A self-installing InstallShield executable (which automatically installs the Tallis Composer and Tester, along with desktop icons, program menu icons, and an uninstaller)
2. A .zip file containing all the relevant classes and resources (which you can then unzip into a folder of your choice).

For best results it may be advisable to uninstall Tallis 1.3 first (Start->Programs->Tallis->Uninstall Tallis 1.3).

Further installation details are available from http://www.openclinical.org/TallisTraining/Documentation_CREDO_2003_017.htm ("Installing the Tallis Authoring Suite")

**Note**

Note that in both cases you will need to have the Java 2 platform (minimum version 1.4) already installed on your computer.

## 2.2 Removing Tallis

If you installed Tallis using the InstallShield executable, then all you need to do is select Start->Programs->Tallis->Uninstall Tallis 1.4. This will remove the whole of Tallis (including all its subfolders) along with any desktop/menu icons.

If you installed Tallis using the manual (.zip) method, then you will need to remove these folders manually.

**Note**

Uninstalling Tallis will remove all its subfolders, so be sure to copy any files you wish to keep to another location first.

# 3. Getting Started

## 3.1 How to get started with Tallis 1.4

You can start using Tallis Composer immediately to do most things. In particular, you can create process descriptions and run them in the Tallis Tester.

Some services, such web enactment or web repository access require prior configuration. Click on the links below for more information about:

Configuring Web Enactment Servers

Configuring Web Repository Servers

Depending on the process descriptions you create, you might need additional information about using Tallis Composer and about the PRO*forma* language. A good place to start is Basic PROforma concepts. Additionally, another resource that can be useful is the Proforma Primer (http://www.openclinical.org/TallisTraining/Documentation_CREDO_2002_008.htm)

## 3.2 Basic PRO*forma* Concepts

### 3.2.1 Overview

Tallis Composer can be used to create and edit *Process Descriptions*, i.e. descriptions, in the PRO*forma* language, of the tasks that are to be carried out by interacting agents (human beings or software components) in order to accomplish some objective.

Process Descriptions can be enacted by the *Tallis Engine*. The engine keeps track of which tasks need to be performed to advance the process, and provides information to external agents regarding the current state of the process. The engine can also receive messages from agents indicating that they have completed certain tasks or provided data relevant to the running of the process.

### 3.2.2 Tasks: the building blocks of the Process Description

Process Descriptions model processes as sets of interacting *tasks*.

Tasks represent activities that are performed by external agents (for instance patients or clinicians) who participate in the enactment of the process, as well as activities performed by the Tallis engine itself without any intervention from external agents.

The first stage in creating a process description is to develop a model of expertise by setting out a collection of tasks that are required in order to achieve a goal. Once the tasks have been composed into a desired network, the details of each task can be filled in.

The 5 types of Tallis tasks are:

| | |
|---|---|
| **Enquiry** | Represents activities that acquire information. |
| | **Example** |

What is the patient’s ethnicity? (Caucasian, Ashkenazi, Other)

**Decision** Represents activities in which a choice is made between several different options.

**Example**

Familial genetic risk assessment:

- HIGH (greater than 25% lifetime risk)
- LOW (less than 17% lifetime risk)
- MEDIUM (17 - 25% lifetime risk)

**Action** Represents simple activities that effect some change to the external world.

**Example**

Reassure and discharge patient

**Plan** A task container - represents “compound” activities that are defined as a set of tasks.

**Keystone** A placeholder task that can later be modified into any of the above.

**Note**

By “later”, we typically mean later in the authoring process, i.e. after the overall process structure has been defined, However, in future releases of Tallis we hope to also support dynamic invocation of process descriptions, so in this context “later” could imply deferring the definition until run time (enactment).

### 3.2.3 The Four Task States

When a Process Description is enacted, tasks generally go through a series of states. All tasks are initially *dormant*; then they are either activated and become *in_progress*, or they are ignored and become *discarded*. Finally, tasks that are performed become *completed*.

A task can be in one of four *States*:

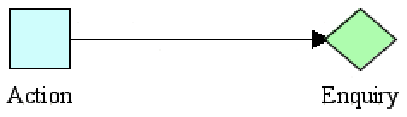
| | | |
|---|---|---|
| ⬛ | **Dormant** | The engine has not yet determined whether the task needs to be performed.<br><br>Task colour: Grey |
| 🟨 | **In_progress** | The task is currently being performed by the engine.<br><br>Task colour: Yellow |
| ⬛ | **Discarded** | The engine has determined that the task does not need to be performed.<br><br>Task colour: Black |
| 🟦 | **Completed** | The task has been performed.<br><br>Task colour: Blue |

The Tallis engine determines task states according to the relationships between the tasks, and according to the task properties.

### 3.2.4 Scheduling Constraints

Scheduling Constraints are a way of specifying the order in which tasks are enacted.

A scheduling constraint is graphically represented by an arrow connecting two tasks. It indicates that the task at the head of the arrow cannot start until the task at the tail of the arrow (the *antecedent task*) has completed.



In the figure above, the Enquiry's antecedent task is the Action. The Action task has to be completed before the Enquiry task can be enacted.

A task can have several antecedent tasks. In order for a task to be considered for enactment, **all** its scheduling constraints must be met – that is, **all** antecedent tasks have to be either *completed* or *discarded*. Consequently, if a task has one or more antecedent tasks that are still *in_progress*, then its scheduling constraints have **NOT** been met (yet).

Once the scheduling constraints have been met, the engine then decides whether a task should become *in_progress* or *discarded*. If **at least one** of the antecedent tasks was *completed*, then the task can become *in_progress**. However, if **all** the antecedent tasks were *discarded*, then the task will be *discarded*.

**Note**

* In fact, the task might still remain dormant if it has other unmet constraints, such as a State Trigger, or Precondition, etc.

### 3.2.5 Task Properties

Task properties are categorised into four groups: general, task specific, scheduling, and execution. This section provides a brief overview of the main task properties.

#### 3.2.5.1 General

Properties common to all tasks:

**Instance Name**

An alphanumeric string representing the ID of the task.

**Example**

PDEnquiry

**Note**

Tallis indexes tasks by Instance Name, i.e. an alphanumeric string chosen by the author. This property is also used to create the task definition, so for this reason, Instance Names must be chosen to be unique (i.e. you cannot create two tasks called “foo” as the definition names will clash). If you want to create a second instance of an existing task, you need to link them.

**Caption**

A textual label that appears in the tree view and graphical view of the Composer; and as the task’s label when executing via the Tallis engine.

**Example**

Patient Details Enquiry

**Goal** A truth-valued expression indicating the intention of the task.

**Example**

blood_pressure < 120

**Note**

In the current implementation of the Tallis engine goals are ignored, but it is hoped that future releases of Tallis will allow authors to specify the conditions under which the goal of a task has (already) been achieved.

**Description** A textual description of the task.

**Example**

This enquiry collects patient demographics

#### 3.2.5.2 Task Specific

Properties specific to the various task types:

**Enquiry: Source** A data item whose value must be supplied.

**Example**

In the Enquiry "Patient Details" a source could be the patient's age, or height, or weight, etc.

**Decision: Candidate** One of a set of options to be selected.

**Example**

In a Decision called "Familial genetic risk assessment", the candidates could be:

- HIGH (greater than 25% lifetime risk)
- LOW (less than 17% lifetime risk)
- MEDIUM (17 - 25% lifetime risk)

**Action: Procedure** An action to be performed.

**Example**

A procedure can be any number of things from a simple message ("reassure and discharge patient"), to instructions for some external system (e.g. update a database).

For more details about task-specific properties, see Decisions, Candidates and Arguments, Enquiries, Sources, and Data Definitions, and Plans.

#### 3.2.5.3 Scheduling

Properties that determine when a task should be executed:

**State trigger** An expression that has to be true before the task can be executed. The task will remain dormant until it becomes true (even if its scheduling constraints have been met).

**Example**

The Action "Control blood pressure" should only be executed if and when the expression "blood_pressure = HIGH" becomes true

**Precondition** An expression that has to be met before the task can be executed.

A task's precondition is examined when, *and only when*, its scheduling constraints are met. If at this moment the precondition is false, the task will be discarded. Otherwise, it becomes *in_progress*.

**Example**

The Enquiry "Check pregnancy history" should only be executed if the expression "patient_gender = female" is true. If it is false (or unknown), then "Check pregnancy history" Enquiry will be *discarded*.

**Note**

If the task should instead *wait* until the specific condition is met, a State trigger should be used.

| | |
|---|---|
| **Event trigger** | A string that enables the end-user to trigger a task during the execution of the Process Description. |

The Event trigger is useful for creating processes that do not have a predefined workflow. During the execution of the Process Description, the end-user will be able to decide which path to take, by triggering the relevant task.

**Example**

A clinician in a patient consultation may choose to invoke one of a number of services, by selecting the relevant Event trigger for each (e.g. "perform_biopsy", or "perform_imaging", or "discharge_patient", etc.)

**Note**

1. The task will not execute unless the trigger is invoked, even if scheduling conditions are met.
2. When the trigger is invoked, the task will execute, even if scheduling conditions are not met.
3. The task state will return to dormant after completion.

#### 3.2.5.4 Execution

Properties that affect the behaviour of a task when it is executed:

| | |
|---|---|
| **Number of cycles** | An integer defining the number of times a task will be repeated. |

**Example**

The action "explain Proforma" may need to be executed 5 times for certain users.

| | |
|---|---|
| **Cycle until** | An expression defining the conditions under which a task will stop cycling. |

**Example**

The action "control_blood_pressure" should continue to cycle until the expression "blood_pressure = normal" becomes true.

| | |
|---|---|
| **Cycle Interval** | An integer defining the time interval between cycles.<br>**Example**<br>The action "administer_drug" should cycle every 4 hours.<br>**Note**<br>The default time unit is hours. |
| **Postcondition** | An assertion (expression) that will be executed when the task has been performed (e.g. a value assigned to a data item).<br>Often used to specify information that will be needed by subsequent tasks.<br>**Example**<br>Once the action "control_blood_pressure" has completed the postcondition "drug_administered = true" is applied. |
| **Optional** | A Boolean specifying whether the task must be completed (or discarded) for its parent plan (i.e. the immediately enclosing plan) to complete.<br>By default, tasks are **not** Optional. This means that their patent plan can complete only if the task is completed or discarded. If a task is Optional, its parent-plan can complete even if the task remains dormant. |
| **Automatic** | A task that is Automatic does **not** require confirmation from the end user. This has different implications on different task types: |

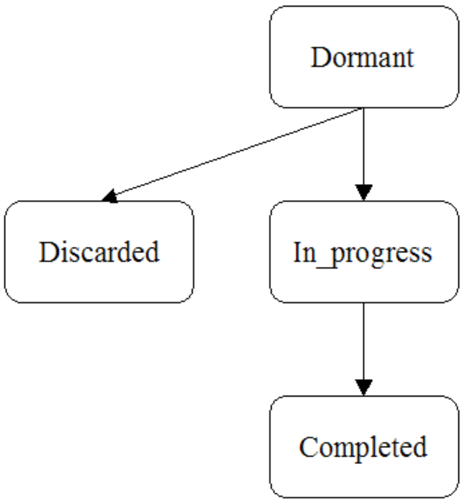
| | |
|---|---|
| **Plan & Enquiry** | The Automatic property is not applicable for Plans and Enquiries |
| **Action** | The Action is confirmed automatically by the engine (the Procedure is not displayed to the end-user). |

| | |
|---|---|
| **Decision** | The recommended Candidate(s) are committed automatically (the Candidates are not displayed to the end-user). |

### 3.2.6 Task State Transitions I

The PRO*forma* engine determines task states according to the scheduling constraints between the tasks, and according to the task properties.

The figure below shows the basic state transitions:



| | |
|---|---|
| **Initial state** | All tasks are initially *dormant*<br><br>A *dormant* task cannot change state until its scheduling constraints are met (Scheduling constraints are met if all the antecedent tasks are either *completed* or *discarded*). |
| **Scheduling constraints are met** | The task will become either *in_progress* or *discarded*, depending on the value of its *precondition* and on the state(s) of its antecedent task(s).<br><br>For it to become *in_progress* the following two conditions must be true:<br><br>• If it has a precondition then that precondition must be met, and |

- If it has any antecedent tasks, at least one is completed.

If either of these conditions is not met, the task becomes *discarded*.

**Note**

In fact, the task might still remain dormant if it has other unmet constraints, such as a State Trigger - see Task State Transitions II.

**Task Completion**

*In_progress enquiries* become *completed* when values have been supplied for their mandatory (i.e. non-optional) sources.

*In_progress actions* become *completed* when they have been confirmed.

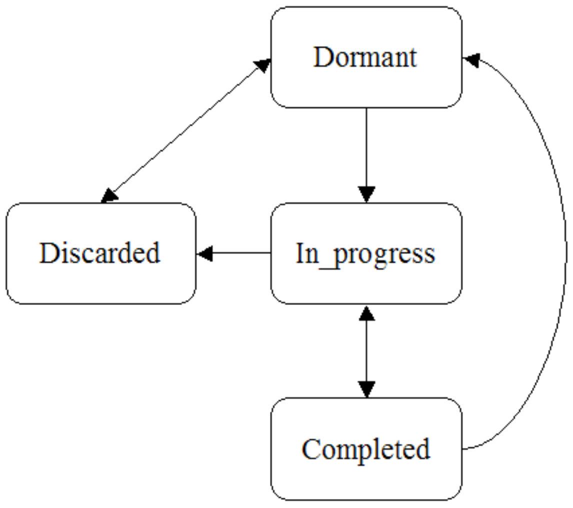
*In_progress decisions* become *completed* when they have been confirmed.

*In_progress plans* become *completed* when all their mandatory (non-optional) sub-tasks have completed.

State transitions are further complicated by other task properties. You can learn more about it in the next section.

### 3.2.7 Task State Transitions II

The diagram below shows all possible state transitions:



However, the precise combination of circumstances under which these transitions actually do occur is quite complex. The best way to understand them is to start with the basic rules given in Task State Transitions I and be aware of the following complications:

- Tasks that are cyclical will return to the dormant state on completion (and then back to *in_progress*, if their Cycle Interval is zero or unspecified)

- Tasks can only become *in_progress* if their parent plan is *in_progress*. This applies also to tasks with Event Triggers.

- Tasks that have been event triggered return to *dormant* on completion, and their parent plan remains *in_progress*. One consequence of this is that any downstream tasks can never be reached, as their antecedents have not all been performed (recall that scheduling constraints are met if only if **all** the antecedent tasks are either *completed* or *discarded*). Other characteristics of event triggers include:
    1. The triggerable task will not execute unless the trigger is invoked, even if scheduling conditions are met.
    2. When the trigger is invoked, the task will execute, even if scheduling conditions are not met.
    3. The task state will return to dormant after completion.

- Tasks with a State Trigger will remain dormant until the state condition becomes true, at which point they follow the usual transition path, i.e. they become *in_progress*, and then after being confirmed they become *completed*. Unlike Event Triggers they do not return to dormant, and hence are not “re-triggerable”. For this reason State Triggers are perhaps better referred to by their original name, i.e. "Wait Conditions".

- Plans can have termination conditions and abort conditions. If a parent plan terminates, it becomes *completed*, and its contents become *discarded*. If a parent plan aborts, it becomes *discarded*, and its contents become *discarded*. Note also that abort conditions are only checked when scheduling constraints have been met, whereas termination conditions are checked in advance.

- An enquiry will complete only when values have been supplied for all its mandatory sources. Any outstanding optional sources will remain requested. The same applies to a decision with sources.

### 3.2.8 Enquiries, Sources and Data Definitions

Enquiries are tasks that collect information from agents (end-users or software components), to be used later in the process. An Enquiry can hold several requests for data, each known as a *Source*.

Each Source is based on a *Data Definition*, which defines the structure of the data that the Enquiry is requesting. A data definition will typically define the following attributes: Name, Caption, Description, Data Type, Range, Default Value, and Selection Mode.

**Note**

- Data Definitions that are not used by Sources are not saved with the Process Description.
- Data Definitions can be saved separately in *Data Definition libraries* (**.dd** files) to facilitate reuse (see Importing and exporting Data Definitions).
- Data Definitions can be imported into a Process Description either from a library, or from another Process Description (see Importing and exporting Data Definitions).

The key difference between a Source and a Data Definition is that a source only exists within the context of an Enquiry and therefore also has a **Status** attribute.

**Note**

Sources also exist within the context of a Decision (see Decisions, Candidates and Arguments).

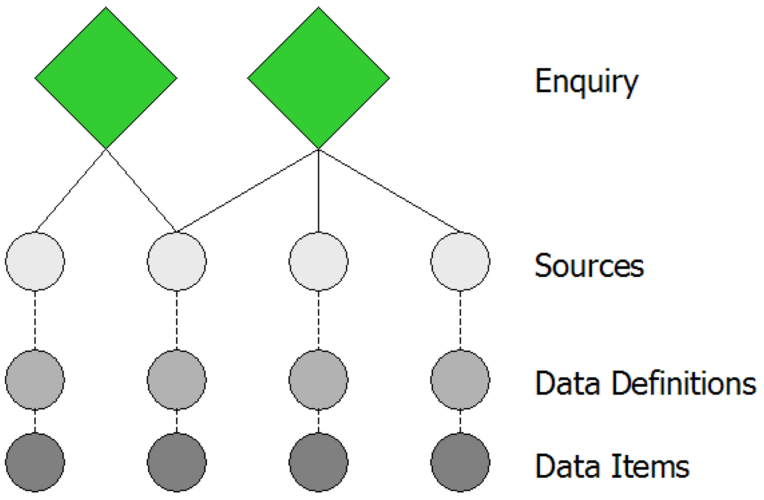
The **Status** of a Source determines whether data entry is required for the Enquiry to complete. By default, Source **Status** is **Mandatory**: this means that the Enquiry will only complete after a data value has been supplied (either by the user, or from some other external agent/system, e.g. a database). If a Source's **Status** is **Optional**, the Enquiry can complete without it.

When a piece of data is collected during runtime enactment, it is held in an instance of a Data Definition, and referred to as a *Data Item*.

**Note**

The current version of Tallis allows for only one instance of a Data Item per Data Definition.

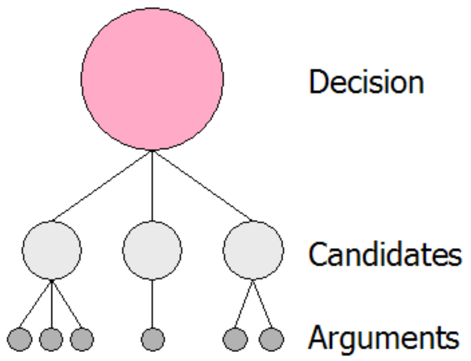
The figure below describes the relationships between Enquiries, Sources, Data Definitions and Data Items: different Enquiries can have a common Source; each Source is based on a singular Data Definition; and one instance of Data Item per Data Definition is formed during runtime enactment.



### 3.2.9 Decisions, Candidates and Arguments

Decisions are tasks in which a choice is made (either by the Tallis Engine or by the end-user) between several different options, known as *Candidates*. PRO*forma* supports decision-making through a mechanism for generating *Arguments* that may be either *for* or *against* a given candidate.

The figure below describes the structure of a Decision: Candidates are a property of a Decision, and Arguments are a property of a Candidate.



An Argument is defined by two components:

| | |
|---|---|
| **Condition** | A truth-valued expression that represents the circumstances under which the argument applies.<br>**Example**<br>blood_pressure < 120 |
| **Support** | The support that the argument offers the Candidate if the condition is true.<br>The support format can be:<br>• Symbolic – the support will be one of the following four categories: the argument can be either **for** or **against** the candidate, or it can **confirm** or **exclude** the candidate<br>• Numeric – the support will be defined using a real value (i.e. the argument has a weight assigned to it) |

The aggregated support of all of a Candidate's Arguments is known as **NetSupport**.

A Candidate's **Decision Rule** is used to decide whether a Candidate will be recommended by the Tallis engine. This rule defines an expression that must return true for the Candidate to be recommended (in practice this usually entails setting a minimum NetSupport threshold).

**Example**

netsupport(some_decision, some_candidate) >= 1

By default, Decisions are **non-automatic**. This means that the Candidates are displayed to the end user, who must then select one of them.

**Note**

- End-users can select non-recommended Candidates if they wish
- One or more candidates may be selectable if the **Candidate Selection** mode is **Multiple Selection**

If several Candidates have the same NetSupport, Candidate **Priority** is used to determine the order in which they will be displayed to the end user. When a Decision is **Automatic**, Candidates are not displayed to the end-user, and the recommended Candidate(s) are committed automatically.

**Note**

Although priority is primarily used to order candidates for display, it *can* sometimes affect which candidate gets committed. For example, in an automatic, single selection decision, if more than one candidate is recommended, the one with the highest priority will be committed.

Another property of Decisions that should be noted is **Sources**. These are requests for data values, and when defined within a Decision, this data has to be collected before the decision can complete (See Enquiries, Sources, and Data Definitions for more information about Sources).

### 3.2.10 Plans

Plans are task containers; they represent the hierarchical structure of a Process Description: each Plan defines a new level in the hierarchy.

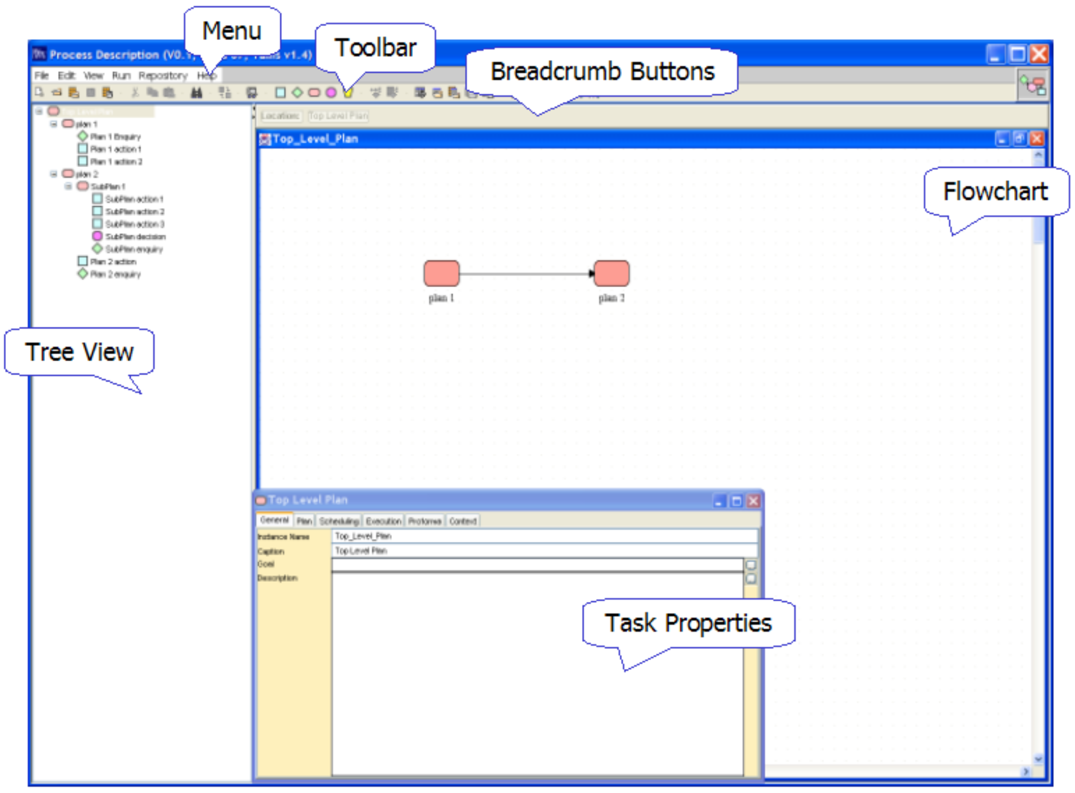
Plans have two unique properties:

| | |
|---|---|
| **Abort Condition** | If this condition is met, the Plan's state is set to **Discarded**. All tasks within the Plan that have not been Completed will be *discarded*, and enactment of the process will halt (i.e., 'downstream' tasks will consequently be *discarded* as well). |
| **Terminate Condition** | If this condition is met, the Plan's state is set to **Completed**. All tasks within the Plan that have not been Completed will be *discarded*, but enactment will continue from the next scheduled task(s) (i.e., tasks 'downstream' from the Plan). |

**Note**

Plans can also affect the scoping of other task properties. For example, a task's event trigger will only be active when its parent plan is in_progress.

## 3.3 Tallis Composer 1.4 screen layout



The figure above is a screen capture of Tallis Composer 1.4.

### 3.3.1 Tree view and flowchart view

Process Descriptions are displayed in Tallis both in a flowchart and in a tree view:

- The **tree view** displays the Process Description's hierarchical structure
- The **flowchart view** displays task ordering according to scheduling constraints

### 3.3.2 Breadcrumb buttons

When you view the contents of a Plan, the Plan caption is maintained as a breadcrumb button, so that you can easily find your way back to higher levels in the Process Description (see Navigating in a Process Description for more information)

### 3.3.3 Task Properties

You can view and edit task properties in the Task Properties window.

To display a task's properties, click on the task in the tree or in the flowchart.

You can hide the Task Properties window by clicking the close button. To re-display it, on the **View** menu, click **Task Properties** (or click the **Show Task Properties** button on the toolbar ).

## 3.4 Navigating in a Process Description

The hierarchy of a Process Description is based on Plans: each Plan defines a new level in the hierarchy.

- You can view all the plans and their contents at a glance in the **tree view**.
- The **flowchart** is more suited for viewing the contents of one Plan at a time.

Double click on a Plan in the tree view to show or hide its contents (i.e., expand & collapse the Plan node).

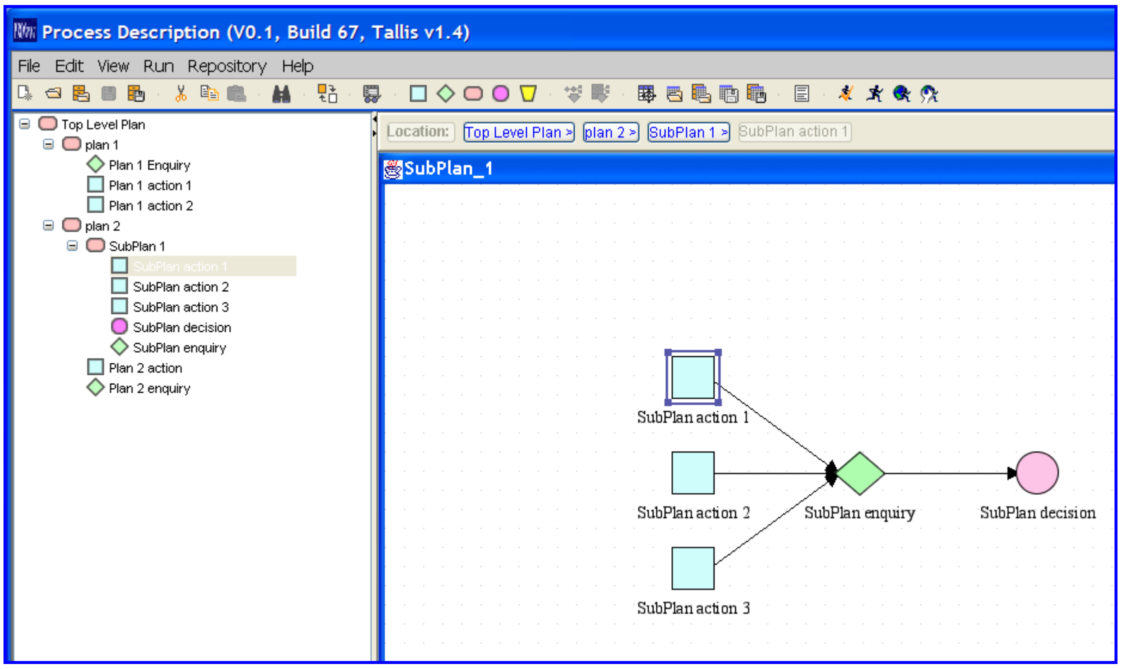
Double click on a Plan in the flowchart to **go down one level** and display the Plan's contents (or click on one of the tasks that are contained within the plan in the tree view).

**To go up a level**, click on the breadcrumb buttons on the panel above the flowchart.

In the figure below, the flowchart displays the contents of 'SubPlan 1'.

The selected task is 'SubPlan action 1'. It is highlighted both in the tree and in the flowchart.

The **breadcrumb buttons** above the flowchart are shortcuts to upper level plans: 'SubPlan 1', 'plan 2', and the 'Top Level Plan'.



# 4. Creating, opening and saving Process Descriptions

### 4.1.1 Creating a new Process Description

On the **File** menu, click **New** (or click the **New** button on the toolbar ).

The Process Description Properties (**Guideline Information**) dialog is displayed. Enter information about the Process Description you are creating and click **OK**, or click **Cancel** to start working on your Process Description straight away. You can enter this information later, by clicking **Properties** on the **File** menu.

### 4.1.2 Opening an existing Process Description

To open a Process Description that is saved locally, on the **File** menu, click **Open** (or click the **Open** button on the toolbar ). Browse and select the Process Description that you would like to open.

### 4.1.3 Loading an existing Process Description from a web repository

To open a Process Description that is saved in a web repository, on the **File** menu, click **Load from Web Repository** (or click the **Load from Web Repository** button on the toolbar ). Browse and select the Process Description that you would like to load.

### 4.1.4 Saving a Process Description

To locally save a Process Description, on the **File** menu, click **Save** (or click the **Save** button on the toolbar ).

To locally save a Process Description that was loaded from a web repository, on the **File** menu, click **Save As**

To save a Process Description to a web repository, on the **File** menu, click **Save to Web Repository** (or click the **Save to Web Repository** button on the toolbar ).

### 4.1.5 Tallis 1.4 file types

When you save a Process Description, four file types are created:

| | |
|---|---|
| **.pf** | The PRO*forma* code |
| **.ui** | Tallis Composer user interface information (for example, the layout of the tasks in the flowchart area) |
| **.md** | Dublin Core meta-data: data entered in the Process Description Properties (**Guideline Information**) dialog |
| **.svg** | An image of the top-level plan: this file is created only when a Process Description is saved to a web repository. The image is displayed as a thumbnail when browsing the repository. |

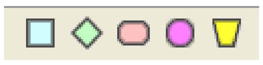
Another file that can be created using Tallis Composer is a *Data Definition library*:

| | |
|---|---|
| **. dd** | Data Definitions can be saved separately in Data Definition libraries to facilitate reuse (see Importing and exporting Data Definitions). |

## 5. Basic Task Manipulations

### 5.1 Inserting a task

Select the task type you want to insert by clicking it on the toolbar.



Click in the flowchart area to place the task.

To change the location of a task, click the task and drag.

Use the Task Properties window to view and edit the task properties.

**Note**

If the Task Properties window isn't displayed, on the **View** menu, click **Task Properties** (or click the **Show Task Properties** button on the toolbar ).

## 5.2 Copying & Pasting a task

Select the task you want to copy by clicking on it.

On the **Edit** menu, click **Copy** (you can also right click on the task and select **Copy** from the pop-up menu, or click the **Copy** button on the toolbar ).

Navigate to the plan in which you want to paste the task.

On the **Edit** menu, click **Paste** (you can also right click in the flowchart and select **Paste** from the pop-up menu, or click the **Paste** button on the toolbar ).

The mouse cursor changes from Arrow to Insert. Click in the flowchart area to place the copied task.

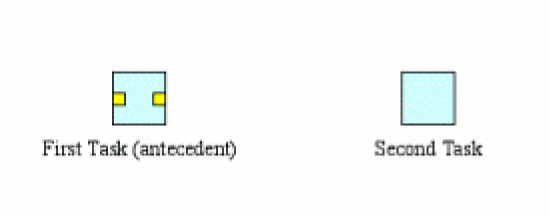
**Note**

You can copy and paste several tasks at a time. To select more than one task, click in the flowchart area and drag over the tasks to form a selection rectangle.

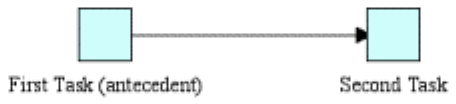
If you copy and paste a task within the same plan, Composer will rename the second instance so that it is unique. See Instance Name in Task Properties

## 5.3 Creating a scheduling constraint

Place the cursor above the first task. Two yellow squares appear on the task.



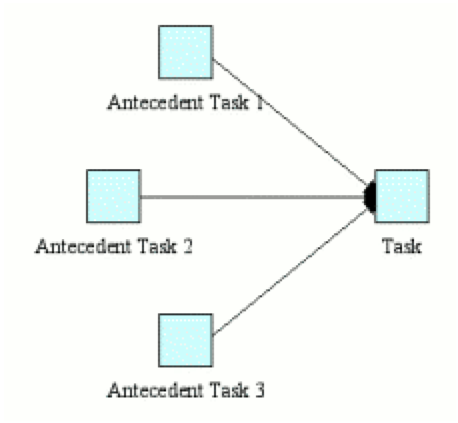
Click on one of the squares, and drag over the second task. Two yellow squares appear on the task – release the mouse key above one of them.



An arrow is formed between the two tasks. The second task will only execute after the first task, i.e., the **antecedent** task, is *completed*. If the antecedent task is *discarded*, the second task will be *discarded* as well.

**Note**

If a task has several antecedent tasks, it will execute only after all of them have been executed **and** if at least one of them has *completed*.



## 5.4 Adding Sources to an Enquiry

Select the Enquiry to which you want to add a Source.

In the Task Properties window, select the Enquiry tab.

**Note**

If the Task Properties window isn't displayed, on the **View** menu, click **Task Properties** (or click the **Show Task Properties** button on the toolbar ).

Use the top panel of the Enquiry tab to add new Sources. The bottom panel is used for managing the Sources of the Enquiry.

### 5.4.1 Creating a Source from Scratch

Sources are based on Data Definitions (see Enquiries, Sources and Data Definitions). When you create a Source from scratch, Composer creates a new Data Definition for it by default.

To create a Source from scratch, enter the following information:

**Name**

An alphanumeric string representing the Source's ID.

**Example**

Symptoms

**Caption**

The caption is displayed to the end-user when the Enquiry is enacted.

**Example**

What are the patient's symptoms?

**Data Type**

The collected data can be of one of four types: **Text**, **Integer**, **Real**, or **Boolean** (yes/no).

**Example**

Symptoms could be of type Text

**Range**

The collected data can be of a limited range of values (i.e., the end-user will have to select one of these predefined values)

**Example**

Symptoms could be: Headache, Nausea, Vomiting, Sensitivity_to_light

**Default Value**

The collected data can have a default value. The default value is used to pre-populate the data entry dialogue box when the Enquiry is enacted.

**Note**

An enquiry will complete when all of its *mandatory* sources have been given values (see **Status**). If at this point there are any outstanding *optional* sources that have not been given values by the user, one of two things will happen:

1. If the optional source has a default value, this will be assigned to the data item, and the enquiry will complete.

2. If the optional source has no default value, the enquiry will still complete, but the source will remain requested. Note that in the Tallis Tester this means that it will remain visible in Requested Data window even though the enquiry has completed. In the Web Enactment interface it remains requested but is no longer visible.

**Selection mode**

When range values of a Source are mutually exclusive, only one of them should be selected. If the end-user has to select one value, selection mode should be set to **single-value**. If the end-user can select more than one value, the selection mode should be set to **multi-value**.

When the Enquiry executes via Web enactment, **single value** selection is displayed as radio-buttons, and **multi-value** selection is displayed as checkboxes. The Tallis Tester currently renders both selection types using radio buttons (when there are four or fewer items in the range), or using a list box (when there are five or more).

**Note**

Multi-selection is only available for Sources with range values.

**Status**

The status of a Source determines the behaviour of the Enquiry during enactment.

When the status is **mandatory**, the Enquiry will remain *In_progress* until all the data is collected, and only then complete.

When the status is **optional**, the Enquiry will complete as soon as all the mandatory Sources are given values.

When all the data is entered, click **Add**.

The new Source will be added to the Source list.

**Note**

All the properties above, except for **Status**, are Data Definition properties. You can view the new Data Definition that was created in the Data Definition Manager.

### 5.4.2 Creating a Source based on an existing Data Definition

Click the button to the right of the Name field. The Data Definition List dialog appears. Select the Data Definition on which you want to base your source, and click **OK**. The Data Definition properties will populate all the fields accept for **Status**. Select the appropriate status, and click **Add**.

The new Source will be added to the Source list.

**Note**

If you modify the Name of the Source before clicking **Add**, a new Data Definition will be created with the modified name.

## 5.5 Adding Candidates to a Decision

Select the Decision to which you want to add Candidates.

In the Task Properties window, select the Decision tab.

**Note**

If the Task Properties window isn't displayed, on the **View** menu, click **Task Properties** (or click the **Show Task Properties** button on the toolbar ).

The Decision tab consists of five sub-tabs. The three tabs that are used for adding Candidates are the Candidates tab, the Arguments tab, and the Decide tab. First select the Candidates tab.

### 5.5.1 Candidates Tab

Use the top panel of the Candidates tab to add new Candidates. The bottom panel is used for managing the Candidates of the Decision.

To create a new source, enter the following information:

| | |
|---|---|
| **Name** | An alphanumeric string representing the Candidate's ID.<br>**Example**<br>ReferToGeneticist |
| **Description** | A text string displayed to the end-user when the Decision is enacted.<br>**Example** |

The patient is eligible for referral to a geneticist

**Note**

There is a small check box labelled ‘Dynamic’. If you find out what this is for, please let us know :)

**Priority** An integer value representing the a-priori importance of the candidate (the larger the value, the higher the importance).

**Note**

If several Candidates have the same NetSupport, Candidate Priority is used to determine the order in which they will be displayed to the end user. See Decision, Candidates and Arguments for more information.

When all the data is entered, click **Add**.

The new Candidate will be added to the Candidate list.

Next, you have to add **Arguments** for (or against) the Candidate. To do this, select the Arguments tab.

### 5.5.2 Arguments Tab

The top drop-down box contains a list of the Decision’s Candidates. Select the Candidate for which you want to define Arguments, and enter the following information:

**Description** A text string displayed to the end-user during enactment (to represent the argument). If there is no description, the condition is displayed instead.

**Note**
There is a small check box labelled ‘Dynamic’. If you find out what this is for, please let us know :)

**Condition** A truth-valued expression that represents the circumstances under which the argument applies.

**Example**

blood_pressure < 120

**Support** The support that the argument offers if the condition is true. The four categories of **symbolic** support are:

| | |
|---|---|
| **For (+)** | If the condition is met, the argument offers some support for the candidate. |
| **Against (-)** | If the condition is met, the argument offers some support against the candidate. |
| **Conf. (++)** | If the condition is met, the argument conclusively confirms the candidate. |
| **Excl. (--)** | If the condition is met, the argument conclusively excludes the candidate. |

In the **Properties** sub-tab of the Decision tab, support type can be modified from **symbolic** to **numeric**. Numeric support allows you to be more precise about the strength of support that an argument offers a candidate, by giving each argument a real-valued weight.

When all the data is entered, click **Add**.

The new Argument will be added to the Arguments list.

### Example

An argument **for** referring a patient to a geneticist would be that the patient has been assessed as being at high genetic risk of breast cancer.

The condition of this argument would be: “Patient’s family genetic risk assessment result” = High

If this condition were true, it would be considered as **support for** the ReferToGeneticist Candidate.

An argument **against** referring a patient to a geneticist would be that the patient has been assessed as being at low genetic risk of breast cancer.

The condition of this argument would be: “Patient’s family genetic risk assessment result” = Low

If this condition were true, it would be considered as **support against** the ReferToGeneticist Candidate.

When a Decision is enacted, the condition and the support properties are used by the engine to determine the effect of each argument and in turn the value of the net support for a given Candidate. A Decision Rule is then used to determine whether or not a particular Candidate is recommended.

To define the Decision Rule for the Candidate, select the Decide tab.

### 5.5.3 Decide Tab

Select the Candidate for which you want to define a Decision Rule from the drop-down box.

The **Rule** field is usually pre-populated with a default decision rule expression, e.g.:

```
netsupport(CurrentDecision, CurrentCandidate)
>= 1
```

This rule states that for this Candidate to be recommended, its aggregated support (or "*netsupport*") has to be greater than or equal to 1.

Selecting **Unlocked** in the **Rule Input Mode** will populate the decision rule with the values in the **NetSupport Value** and the **NetSupport Operator** fields (the fields are disabled while **Locked** is selected).

Modifying the **NetSupport Value** and the **NetSupport Operator** in the **Unlocked** mode makes these values available for the other Candidates of the Decision.

# 6. Advanced Task Manipulations

## 6.1 Linking a task

### 6.1.1 What are linked tasks?

Linked tasks share a common task *definition*, and therefore the values of most task properties. Consequently, when the properties of one task are modified, the other task's properties are updated accordingly. However, certain properties are associated with task *instances*, and they can have different values:

- Cycle properties (cycle number, cycle until, cycle interval)
- Optional
- Automatic

These properties can be found on the **Execution** tab of the **Task Properties** window.

**Note**

- You can only link tasks from different plans - tasks that belong to the same plan cannot be linked.

- You can only link tasks of the same type.

Linking is best applied when you want to access the same task functionality from more than one place in your process description.

### 6.1.2 How do you link tasks?

To link one task to another, select the task you want to link.

**Note**

The values of the task properties of this task will be replaced with the values of the task properties of the task you will link to.

On the **Edit** menu, click **Link to Task** (you can also click **Link to Task** on the pop-up menu, or click the **Link to Task** button on the toolbar )

A dialog appears containing all linkable tasks. Select the task that you want to link to, and click **OK**. The task's name will be set so that both tasks now have the same name.

**Note**

You can create a copy of an existing task without sharing task properties by using Copy & Paste.

## 6.2 Inserting a Process Description as a plan

You can insert an existing Process Description into the Process Description you are currently working on. The existing Process Description will be inserted as a Plan.

Insert a Keystone into your current Process Description.

**Note**

This keystone will eventually be replaced by a plan, which will contain the inserted Process Description.

To insert a locally saved Process Description, click the **Load Local Process Description As Component** button on the toolbar (or right click and select **Local Import** from the pop-up menu).

To insert a web repository Process Description, click the **Load Repository Process Description As Component** button on the toolbar (or right click and select **Web Repository Import** from the pop-up menu).

Select the requested Process Description, and click **Open** (or **Load**).

The keystone should be replaced with a Plan that takes its name from the top-level plan of the inserted Process Description.

**Note**

An error message will appear if there are conflicts between the current Process Description you are working on and the Process Description you are trying to insert, such as identical task names in the two processes. You need to resolve any such conflicts before inserting the Process Description.

# 7. Enacting A Process Description

## 7.1 Verifying a Process Description

You can search for syntactic and basic semantic errors in your Process Description by using the **Verify** tool.

On the **Run** menu, click **Verify** (or click the **Verify** button on the toolbar ).

The **Process Description Verification Report** dialog appears. In the upper panel is a list of all the tasks containing errors. The lower panel displays error details for the selected task. When you select a task in the dialog, it is also selected in the work area (tree & flowchart). The report dialog is modeless; you can keep it open while you make the necessary changes to your Process Description, and then click **Update** to re-verify the Process Description.

**Note**

The **Cancel** button closes the report dialog.

## 7.2 Running a Process Description in the Tester

You can enact a Process Description locally by running it in the Tallis Tester.

On the **Run** menu, click **Run In Tester** (or click the **Run In Tester** button on the toolbar ).

The Process Description will first be verified. If no errors are found, the Tallis Tester will be launched, and the Process Description will be enacted.

**Note**

UI modifications such as customised web pages do not appear in the Tallis Tester.

Also note that due to the Web Enactment being based around a sequential set of web pages (rather than a set of interactive panels as in the Tester) there are certain other minor differences in the runtime behaviour (most notably concerning the relationship between sources and enquiries).

## 7.3 Running a PD in a Web Browser

You can enact a Process Description in a web browser by submitting it to a web enactment server.

If you have one or more enactment servers set up, you can run the current Process Description on the default enactment server by clicking **Run in Web Browser (Default Enactment Servers)** on the **Run** menu (or by clicking the **Run in Web Browser** button on the toolbar ).

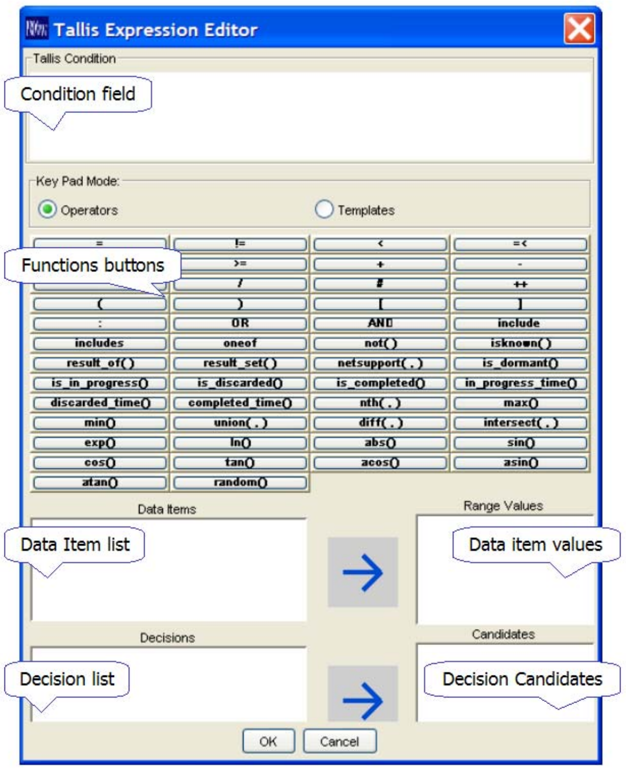
To select between multiple enactment servers, click **Run In Web Browser…** on the **Run** menu. The **Enactment Servers** dialog appears. From the drop-down list box, select the server configuration with which you want to run the Process Description, and click **Enact**.

Click here to learn more about Configuring Web Enactment Servers.

# 8. Using the Expression Editor

The Expression Editor is accessible from every field in the task properties window that contains a condition or an expression (e.g., State Trigger, Precondition, Cycle Until). To open it, click on the ellipsis (**…**) button to the right of the field.

The figure below is a screen capture of the Expression Editor:



Type your expression in the **Condition** field.

You can add **functions** to the expression by double-clicking the function buttons. Function buttons have two modes:

1. Operators – double-clicking a button inserts a function into the expression
2. Templates – double-clicking a button inserts a template of the function into the expression. The template includes both function and placeholders for parameters and values.

   The placeholders and what they stand for:

| | |
|---|---|
| **X** | Expression |
| **T** | Task |
| **D** | Decision |
| **C** | Candidate |
| **S** | Set |

In both modes, double-clicking a function button inserts the button label into the expression.

You can add **Data Items** or **Data Item Values** to the expression by double-clicking a selected item. The Range Values list displays the values of the selected Data Item.

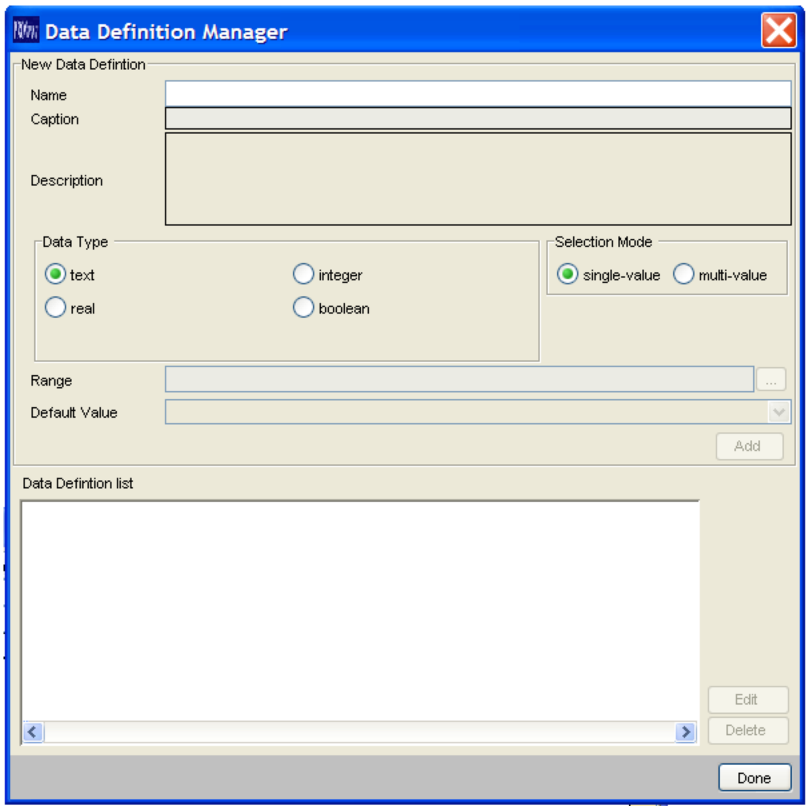
You can add **Decisions** or **Candidates** to the expression by double-clicking a selected item. The Candidates list displays the Candidates of the selected Decision.

# 9. Using the Data Definition Manager

The Data Definition Manager can be used to manage the Data Definitions of the current Process Description. It has the same basic structure as the Enquiry tab of the Task Properties window, but while the Enquiry tab displays a list of all Sources relevant to a given enquiry, the Data Definition Manager displays a list of all the Data Definitions in the complete Process Description.

To access the Data Definition Manager, on the **View** menu, click **Process Data Definitions** (or click the **Data Definition Manager** button on the toolbar ).

The figure below is a screen capture of the Data Definition Manager:



The top panel is used for adding new Data Definitions (see Adding Sources to an Enquiry for more information).

The lower panel displays the Data Definitions. Use the **Edit** and **Delete** buttons to the right of the list to manage the Data Definitions.

**Note**

- Data Definitions that are not used by Sources are not saved with the Process Description.
- Data Definitions can be saved separately in *Data Definition libraries* (**.dd** files) to facilitate reuse (see Importing and exporting Data Definitions).
- Data Definitions can be imported into a Process Description either from a library, or from another Process Description (see Importing and exporting Data Definitions).

## 10. Configuring Web Enactment Servers

On the **Run** menu, click **Configure Web Enactment Servers** (or click the **Configure Web Enactment Servers** button on the toolbar ).

The **Enactment Servers Configuration** dialog appears. Click Add to add a new sever configuration.

In the **Add Server Configuration** dialog, enter the server **URL**, and the enactment **Application**.

**Example**

| | |
|---|---|
| URL: | www.openclinical.org |
| Application: | newkpc |

Click **OK** to confirm the **Add Server Configuration** dialog.

**Note**

The first server configuration that is added automatically becomes the default server configuration. It will be the configuration used when running a Process Description in a web browser. You can make any configuration the default one by selecting it and clicking **Set as Default**.

Click **OK** to confirm the **Enactment Servers Configuration** dialog.

## 11. Configuring Web Repository Servers

You can use a web repository server to load or save process descriptions.

On the **Repository** menu, click **Configure Repository Servers**.

The **Repository Servers Configuration** dialog appears. Click ‘Add’ to add a new sever configuration.

In the **New Repository Server Configuration** dialog, enter the **Server URL** and the **Repository Application**. You can enter the **User Name** and the **Password** now, or when you first log in.

**Example**

| | |
|---|---|
| Server URL: | www.openclinical.org |
| Repository App.: | newRepositoryService |
| Enactment App.: | kpc |
| User Name: | Guest |
| Password: | ******** |

Click **OK** to confirm the **New Repository Server Configuration** dialog.

**Note**

The first server configuration that is added automatically becomes the default server configuration. It will be the selected configuration in the drop-down list box of the Repository Login dialog. You can make any configuration the default one by selecting it and clicking **Set as Default**.

Click **OK** to confirm the **Repository Servers Configuration** dialog.

# 12. Using the Repository Explorer

The Repository Explorer can be used to manage Repository files and folders, and to enact Repository Process Descriptions on a server.

To access the Repository Explorer, on the **Repository** menu, click **Explorer**.

The figure below shows the Repository Explorer. The left panel contains a list of the files and folders in the Repository. This list can be filtered by the **View Mode** radio-buttons, to display:

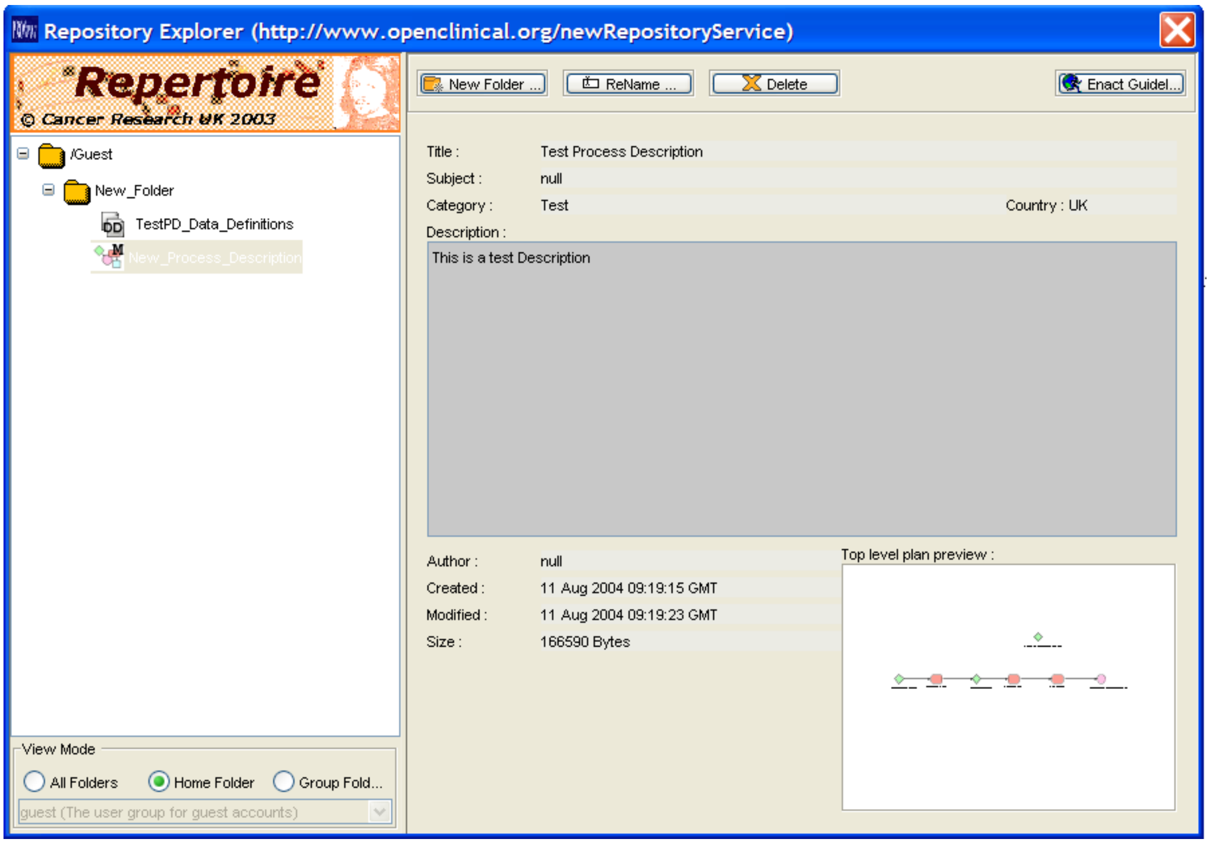
- All Folders
- Home Folders – the folder you were assigned when your account was created
- Group Folders – select a group from the drop-down list to display folders of users that belong to that group

Object nodes in the Repository tree  can be of two types:

- Process Descriptions
- Data Definitions

When you select a Process Description file, the file details are displayed in the right panel.

To enact a Process Description, select it and click **Enact Guidel…**. Use the **New Folder**, **Rename** and **Delete** toolbar buttons to manage files and folders.



Title : Test Process Description
Subject : null
Category : Test
Country : UK
Description :
This is a test Description
Author : null
Created : 11 Aug 2004 09:19:15 GMT
Modified : 11 Aug 2004 09:19:23 GMT
Size : 166590 Bytes

## 13. Using the Tallis Tester

The Tallis Tester is a stand-alone application that allows you to test the flow of your Process Description, by enacting it locally.

**Note**

UI modifications such as customised web pages do not appear in the Tallis Tester.

Also note that due to the Web Enactment being based around a sequential set of web pages (rather than a set of interactive panels as in the Tester) there are certain other minor differences in the runtime behaviour (most notably concerning the relationship between sources and enquiries).

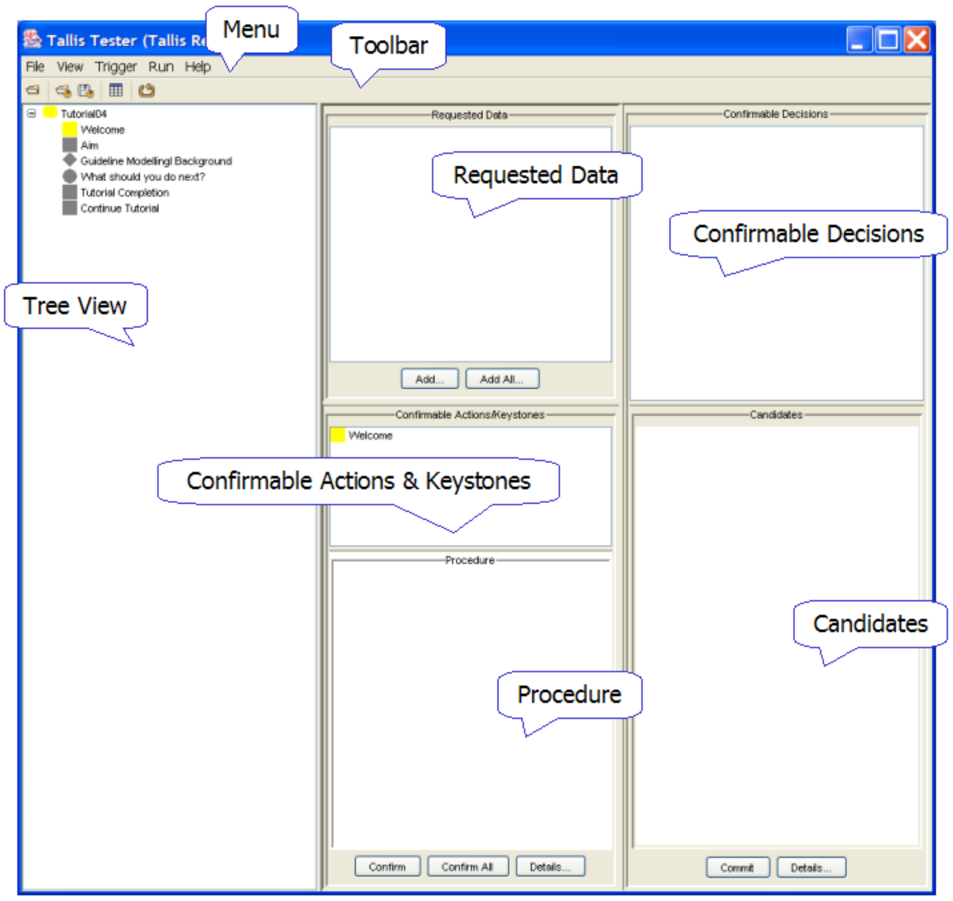
Tallis Tester can be launched from Tallis Composer: on the **Run** menu, click **Run In Tester** (or click the **Run In Tester** button on the toolbar ).

The Process Description will first be verified. If no errors are found, Tallis Tester will be launched, and the Process Description will be enacted.

**Note**

When the Tallis Tester is running, you can load and enact Process Descriptions by clicking **Load Process** on the **File** menu (or by clicking the **Load a PROforma Process Description** button on the toolbar ).

The figure below is a screen capture of the Tallis Tester's main screen:



## 13.1 Tree View

The Process Description is displayed in a tree view, similar to the one in Tallis Composer. The colours of the tasks change during the enactment, as they represent the tasks' states:

| | | |
|---|---|---|
| ⬛ | **Dormant** | The engine has not yet determined whether the task needs to be performed.<br>Task colour: Grey |
| 🟨 | **In_progress** | The task is currently being performed by the engine.<br>Task colour: Yellow |

| | | |
|---|---|---|
| 🟨 | **In_progress Decision with a recommended Candidate** | In_progress Decisions can be either yellow or orange:<br>• Yellow: There are no recommended Candidates for this Decision<br>• Orange: There is at least one recommended Candidate for this Decision |
| ⬛ | **Discarded** | The engine has determined that the task does not need to be performed.<br>Task colour: Black |
| 🟦 | **Completed** | The task has been performed.<br>Task colour: Blue |

**Note**

Note that in Tallis Composer task colours represent the task type, and are therefore different from the task colours in the Tester.

## 13.2 Confirmable Tasks and Requested Data

During enactment, the panels of the Tester are populated with In_progress tasks that require end-user confirmation (i.e., are non-automatic).

### 13.2.1 Actions and Keystones

- Actions and Keystones are displayed in the **Confirmable Actions/Keystones** panel.
- The Procedure of the selected action is displayed below, in the **Procedure** panel.
- Confirm an Action (or Keystone) by selecting it and clicking **Confirm** (you can select multiple tasks by using the CTRL and SHIFT Keys; you can confirm all tasks in the panel by clicking **Confirm All**).
- Click **Details** to view further information about the selected task.

### 13.2.2 Decisions

- Decisions are displayed in the **Confirmable Decisions** panel.

**Note**

A Decision that has Sources defined in its Sources tab will only become confirmable after the data for the Mandatory Sources has been collected. Thus, a Decision might be In_progress (as indicated by its yellow colour in the tree view), but not confirmable (and therefore it would not appear in the Confirmable Decisions panel).

- The Candidates of the selected Decision are displayed below, in the **Candidates** panel.
    - Commit a Candidate by selecting it and clicking **Commit** (if the **Candidate Selection** mode of the selected Decision is **Multiple Selection**, you can commit multiple candidates; you can select multiple Candidates by using the CTRL and SHIFT Keys).
    - Click **Details** to view further information about the selected Decision.

#### 13.2.3 Sources

- When an Enquiry (or a Decision) becomes In_progress, its Sources are displayed in the **Requested Data** panel.
- To enter data, select a Source and click **Add** (you can enter data for all the Sources in the panel by clicking **Add All**).

## 13.3 Task Details

The task details dialog displays information about the task:

- Common task attributes – Name, Caption, State, current Cycle number (out of how many cycles), Parameters, and Description.
- For Actions, the Procedure is displayed.
- For Decisions, Candidates are displayed (green if they are recommended, red if not), as well as the Arguments for the selected Candidate (black if the argument's condition is true, grey if it's false).

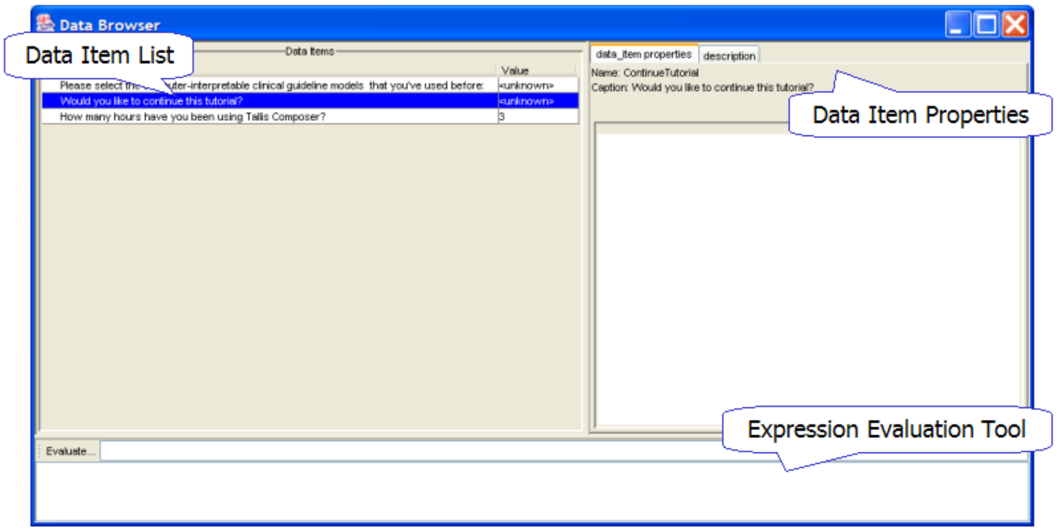
To access the Task Details dialog, double-click on a task (either in the tree view or in one of the panels) or select a confirmable task in one of the panels and click **Details**.

## 13.4 Data Browser

The Data Browser dialog displays a list of the Data Items and their current values, as well as the properties of the selected Data Item.

It also includes an expression evaluation tool, which enables you to evaluate the current value of an expression.

Type the expression you want to evaluate in the field to the right of the **Evaluate** button, and click **Evaluate** (or ENTER). The result of the evaluation will be displayed in the field below.



Data Items

| | Value |
|---|---|
| Please select the [illegible]uter-interpretable clinical guideline models that you've used before: | <unknown> |
| Would you like to continue this tutorial? | <unknown> |
| How many hours have you been using Tallis Composer? | 3 |

data_item properties | description

Name: ContinueTutorial
Caption: Would you like to continue this tutorial?

To access the Data Browser dialog, click **Data Browser** on the **View** menu (or click the **Browse Data and Evaluate Expressions** button on the toolbar ).

## 13.5 Saving and Loading Process Description States

At any point in the enactment of a Process Description, you can save the Process Description state, that is, the stage to which it got in the enactment. Later, you will be able to continue the enactment from that stage. The Process Description states are saved as **.pfs** files.

To save a Process Description state, on the **File** menu, click **Save State** (or click the **Save the Current Process State to a File** button on the toolbar ).

To restore a Process Description state, on the **File** menu, click **Restore State** (or click the **Restore Process State from a File** button on the toolbar ).

## 13.6 Tallis Tester Toolbar

| | | |
|---|---|---|
| | **Load a PROforma Process Description** | Load and enact a Process Description |
| | **Restore Process State from a File** | Load a Process Description with a saved state |
| | **Save the Current Process State to a File** | Save the current state of the Process Description |
| | **Browse Data and Evaluate Expressions** | Browse Data Item values and evaluate expressions |
| | **Restart the Current Process** | Restart the enactment of the current Process Description |

# 14. Advanced Features

## 14.1 Importing and exporting Data Definitions

### 14.1.1 Exporting Data Definitions

Data Definitions can be saved separately in *Data Definition libraries* (**.dd** files) to facilitate reuse.

To save Data Definitions locally, on the **File** menu, under **Save Data Definitions**, click **Locally** (or click the **Save Data Definitions** button on the toolbar ).

To save Data Definitions to a web repository, on the **File** menu, under **Save Data Definitions**, click **To Web Repository** (or click the **Save Data Definitions to Web Repository** button on the toolbar ).

Enter a name for the Data Definition library, and click **Save**.

**Note**

Data Definitions that are not used by Sources are not saved with the Process Description.

#### 14.1.2 Importing Data Definitions

Data Definitions can be imported into a Process Description for reuse either from a library, or from another Process Description.

To Load locally saved Data Definitions, on the **File** menu, under **Load Data Definitions**, click **Locally** (or click the **Load Data Definitions** button on the toolbar ).

To Load Data Definitions from a web repository, on the **File** menu, under **Load Data Definitions**, click **From Web Repository** (or click the **Load Data Definitions from Web Repository** button on the toolbar ).

Select a Data Definition library (**.dd** file) or a Process Description (**.pf** file), and click **Open** (when importing locally) or **Load** (when importing from a web repository).

All the Data Definitions from the library or the Process Description will be available for use in your current Process Description. You can view the Data Definitions in the Data Definition Manager.

### 14.2 Customising the appearance of web enactment

The appearance of web enactment can be customised to support additional functionality (e.g. client-side scripting, or the inclusion of additional content such as bespoke HTML pages, images, video, etc.). More details on how to do this are available from http://www.openclinical.org/TallisTraining/Documentation_CREDO_2005_001.htm (“Tallis Interface Primer”).

### 14.3 Working with external databases

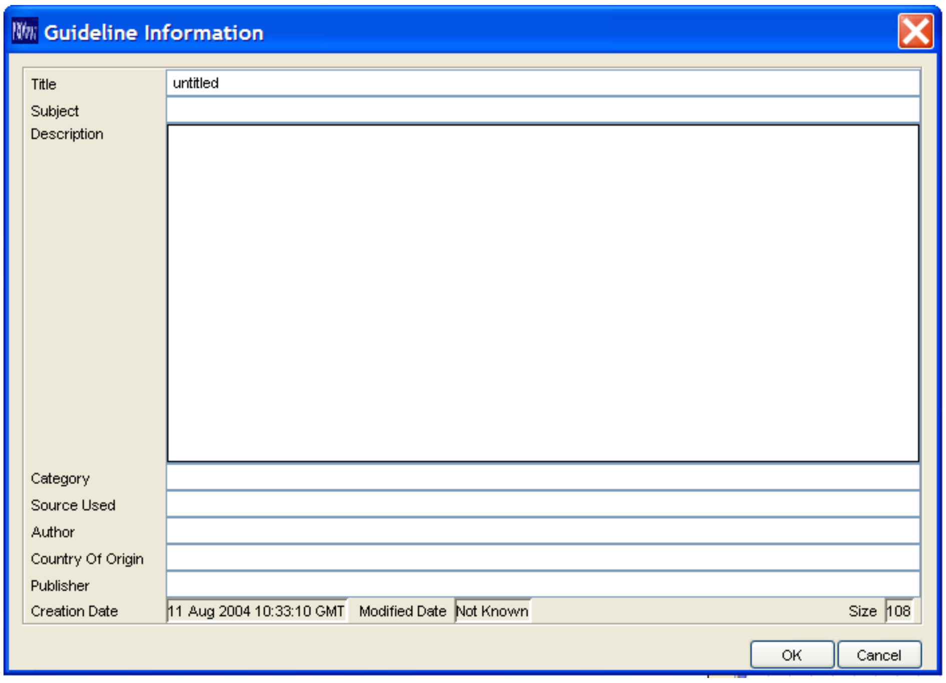
Tallis can be made to interact with other data sources (e.g. databases) by using specific XML configuration files. Further details on how to do this are available from http://www.openclinical.org/TallisTraining/Documentation_CREDO_2004_006.htm (“Generic Data Access for Tallis”).

# 15. Tallis Composer Dialogs

### 15.1 Process Description Properties

The Process Description Properties (**Guideline Information**) dialog opens automatically when you create a new Process Description. Its aim is to give users general information about the Process Description (Title, Subject, Description, Category, Source Used, Author, Country Of Origin, Publisher).

Additionally, this dialog provides some technical information (Creation Date, Modified Date, Size).



| | |
|---|---|
| Title | untitled |
| Subject | |
| Description | |
| Category | |
| Source Used | |
| Author | |
| Country Of Origin | |
| Publisher | |
| Creation Date | 11 Aug 2004 10:33:10 GMT |
| Modified Date | Not Known |
| Size | 108 |

**Note**

You can access the Process Description Properties by clicking **Properties** on the **File** menu.

## 15.2 Working with the Task Properties window

When you select a Task in the Process Description, the values of its properties are displayed in the Task Properties window.

**Note**

If the Task Properties window isn't displayed, on the **View** menu, click **Task Properties** (or click the **Show Task Properties** button on the toolbar ).

The title bar displays the Task's **Type** and **Instance Name**.

Task properties are categorised into four groups, each displayed on a separate tab: general, task specific, scheduling, and execution.

### 15.2.1 General tab

Properties that differentiate the task from other tasks of the same type

| | |
|---|---|
| **Instance Name** | An alphanumeric string representing the ID of the task. |

| | |
|---|---|
| **Caption** | A textual label that appears in the tree view and graphical view of the Composer; and as the task's label when executing via the Tallis engine. |
| **Goal** | A truth-valued expression indicating the intention of the task. |
| **Description** | A textual description of the task. |

### 15.2.2 Task Specific tab

Properties that differentiate between tasks of different types

**Action**

| | |
|---|---|
| **Procedure** | Defines the detailed action to perform.<br><br>The **Trigger List** displays Event Triggers of other tasks in the same parent-plan. You can create a hyperlink that will in runtime enact one of these tasks by invoking its Event Trigger: Type a text string in the edit-box, select it, and then select the appropriate trigger.<br><br>The View Mode can be either Plain Text (for entering ASCII text) or HTML (for entering HTML via a WYSIWYG editor). |

**Enquiry**

| | |
|---|---|
| **Sources** | Define the data items for which values must be supplied. Use this tab to create Sources by linking to existing Data Definitions, or by creating new ones.<br><br>For more information, see Adding Sources to an Enquiry. |

## Decision

| | |
|---|---|
| **Candidates tab** | Candidates define the options to choose between. Use this tab to enter the Names, Descriptions, and Priorities of the Decision Candidates.<br><br>For more information, see Adding Candidates to a Decision. |
| **Sources tab** | Sources defined in this tab must be supplied with values before the Decision is enacted.<br><br>For more information about creating Sources, see Adding Sources to an Enquiry. |
| **Arguments tab** | PRO*forma* supports decision-making through a mechanism for generating Arguments for and against Candidates. Use this tab to define Arguments for each Candidate.<br><br>For more information about creating Arguments, see Adding Candidates to a Decision. |
| **Decide tab** | When a Decision is enacted, the condition in the Decision Rule field determines whether or not a Candidate is recommended.<br><br>For more information, see Adding Candidates to a Decision. |
| **Properties tab** | **Support Type** determines the structure of the **Support** field in the **Aguments tab**.<br><br>• **Symbolic** displays four categories of support (the argument can be either for or against the candidate, or it can confirm or exclude the candidate).<br><br>• **Numeric** displays a numeric weight field that allows for a more specific definition of support (the argument has a weight assigned to it).<br><br>**Candidate Selection** determines whether only one Candidate (**Single Selection**) or more than one (**Multiple Selection**) can be selected during runtime. |

**Plan**

| | |
|---|---|
| **Abort Condition** | A truth-valued expression indicating the conditions under which the task should abort. Once this condition is met, the Plan's state is set to **Discarded**. All tasks within the Plan that have not been Completed will be discarded as well. |
| **Terminate Condition** | A truth-valued expression indicating the conditions under which the task should terminate. Once this condition is met, the Plan's state is set to **Completed**. All tasks within the Plan that have not been Completed will be discarded. |

**Keystone**

As it is a temporary placeholder, the Keystone doesn't have task-specific properties.

### 15.2.3 Scheduling tab

Properties that determine when a task should be enacted

| | |
|---|---|
| **State trigger** | A truth-valued expression indicating a condition that has to be met before the task can be enacted. As long as it isn't met, the task will remain dormant. |
| **Event trigger** | An alphanumeric string that enables the end-user to trigger a task during the enactment of the Process Description.<br><br>The Event Trigger is useful for creating processes that are not linear. During the enactment of the Process Description, the end-user will be able to decide which path to take, by triggering the relevant task. |
| **Precondition** | A truth-valued expression indicating a condition that has to be met before the task can be enacted. A task's precondition is examined when, *and only when*, its scheduling constraints are met. If, at this moment the precondition is false, the task will be discarded. |

### 15.2.4 Execution tab

Properties that affect the enactment scheme of the task

| | |
|---|---|
| **Cycle until** | An expression defining the conditions under which a task will stop cycling. |
| **Cycle Interval** | An integer defining the time interval between cycles.<br><br>**Note**<br>The default time unit is hours. |
| **Number of cycles** | An integer defining the number of times a task will be repeated. |
| **Optional** | A task that is Optional isn't necessary for the completion of the guideline.<br><br>By default, tasks are not Optional. This means that their patent-plan can complete only if the task is completed or discarded. If a task is Optional, its parent-plan can complete even if the task is dormant.<br><br>**Valid Values**<br>True or False |
| **Automatic** | A task that is Automatic doesn't request end-user confirmation. This property is only applicable to Actions and Decisions.<br><br>**Valid Values**<br>True or False |
| **Postcondition** | An assertion (expression) that will be executed when the task has been performed (e.g. assigning a value to a data item). Often used to specify information that will be needed by subsequent tasks. |

### 15.2.5 PRO*forma* tab

Displays the task properties in PRO*forma* code.

**Note**

- The PRO*forma* tab is read-only.

### 15.2.6 Context Tab

The context tab is used to define parameters that provide further control over the enactment process (e.g. configuration parameters for data access, web enactment user interface, etc.)

## 15.3 Web Repository Load & Save

The **Load from Web Repository** and the **Save to Web Repository** dialogs have the same structure as the Repository Explorer.

### 15.3.1 Loading a Process Description

Either select the requested Process Description from the list on the left, or type in its name in the **Application Name** field; click **Load**.

### 15.3.2 Saving a Process Description

Select the folder in which you want to save the Process Description. You can modify the name under which the Process Description will be saved in the **Application Name** field. Click **Save**.

## 15.4 Process PRO*forma* Definition

The Process PRO*forma* Definition dialog displays the Process Description in PRO*forma* code.

To access the Process PRO*forma* Definition dialog, on the **View** menu, click **Process PRO*forma* Definition** (or click the **Process PRO*forma* Definition** button on the toolbar ).

**Note**

- The PRO*forma* code is read-only.
- To view the PRO*forma* code of a specific task, select the task and select the **PRO*forma*** tab in the **Task Properties** window.

To close the dialog, click **Cancel**.

# 16. Tallis Composer Toolbar

| | | |
|---|---|---|
| | **New** | Create a new Process Description |
| | **Open** | Open a locally saved Process Description |
| | **Load from Repository** | Load a Process Description from a Web Repository |
| | **Save** | Save a Process Description locally |
| | **Save to Repository** | Save a Process Description to a Web Repository |
| | **Cut** | Move the selected item(s) to the Tallis Clipboard |
| | **Copy** | Copy the selected item(s) to the Tallis Clipboard |
| | **Paste** | Paste item(s) from the Tallis Clipboard |
| | **Search** | Search for an alphanumeric string |
| | **Link to Task** | Create a second instance of an existing task |
| | **Show Task Properties** | Display the Task Properties window |
| ◻ | **Action** | Insert an Action task |
| ◇ | **Enquiry** | Insert an Enquiry task |
| ▭ | **Plan** | Insert a Plan task |
| ● | **Decision** | Insert a Decision task |
| ▽ | **Keystone** | Insert a Keystone task |
| | **Load Local Process Description As Component** | Insert a locally saved Process Description to replace a Keystone |

| | | |
|---|---|---|
| | **Load Repository Process Description As Component** | Insert Process Description from a Web Repository to replace a Keystone |
| | **Data Definition Manager** | View & manage all Data Definitions |
| | **Load Data Definitions** | Import Data Definitions from a local Data Definition library or from a local Process Description |
| | **Load Data Definitions from Web Repository** | Import Data Definitions from a Web Repository Data Definition library (the current version of Tallis doesn't import Data Definitions from a Web Repository Process Description). |
| | **Save Data Definitions** | Save Data Definitions to a local Data Definition library (.dd file) |
| | **Save Data Definitions to Web Repository** | Save Data Definitions to a Web Repository Data Definition library (.dd file) |
| | **View PRO*forma* Definition** | Display the PRO*forma* code of the Process Description (read-only) |
| | **Verify** | Search the Process Description for syntactic and basic semantic errors |
| | **Run in Tester** | Local enactment of the Process Description by the Tallis Engine |
| | **Run in Web Browser** | Web enactment of the Process Description by the Tallis Engine on a Server |
| | **Configure Web Enactment Servers** | Define configurations of Servers and enactment applications for web enactment of Process Descriptions |