Project Number	IST-2006-033789
Project Title	Planets
Title of Deliverable	Guidelines for Creating and Installing IF Preservation Workflows and Templates
Deliverable Number	IF5-D1
Contributing Sub-project and Work-package	IF5
Deliverable Dissemination Level	External Public
Deliverable Nature	Report
Contractual Delivery Date	31st May 2009
Actual Delivery Date	19th June 2009
Author(s)	ARC

Abstract

The document describes the API and functionality of the Planets workflow execution engine (WEE) version 2. We provide an overview of the main components, the public interfaces, and describe how it is utilized through the Workflow Control Panel (WCP). The document specifically provides user documentation for configuring, deploying, and accessing Planets workflow templates.

Keyword list:

Interoperability Framework, Workflow Execution Engine (WEE), Template Repository, Workflow Templates, Workflow Control Panel (WCP)

Contributors

Person	Role	Partner	Contribution
Ross King	IF SP Lead	ARC	Workflow Control Panel report
Andrew Lindley	IF/TB Developer	ARC	Workflow Execution Engine report
Rainer Schmidt	IF5, IF7 Lead	ARC	Editor

Document Approval

Person	Role	Partner
Ross King	IF SP Lead	ARC
Christen Hedegaard	PA/4 WP Lead	KB-DK

Distribution

Person	Role	Partner

Revision History

Issue	Author	Date	Description
0.1	RS	12/06/2009	Initial draft version for internal review
0.2	RK	14/06/2009	Corrections from internal review
0.3	CH	19/06/2009	External Review
1.0	RK	19/06/2009	Final version

References

Ref.	Document	Date	Details and Version

TABLE OF CONTENTS

1. Introduction

The document provides an overview of the Planets workflow environment version 2, as well as a user guide to creating and utilizing custom workflow templates.

The employment of a suitable workflow orchestration system imposes a crucial aspect for the Planets environment. Preservation experiments are comprised of a number of well defined execution steps, which have to be executed within a service-oriented distributed environment. Such preservation processes involve complex control logic as well as web service interactions and data model manipulations.

A preservation workflow consists of a sequence of activities, carried out in a specific order, in which the output parameters of one action are validly mapped to the input parameters of the following action. An example of a Planets workflow would be: for a given file, first identify a file format, then validate the format, then characterize it, then migrate it to a new format, then characterize the new file, and finally compare with the original.

Initially, we examined the application of the Web Service Business Process Execution Language (WS-BPEL) and implemented experimental preservation workflows based on WS-BPEL. This approach provided a large degree of flexibility and allowed the specification of preservation processes at a low (i.e. messaging) level using Web service standard languages like BPEL, XML, XPath, and WSDL. However, work in this direction was hindered by two difficulties; first, that the BPEL language is very powerful but also low-level and hence very complex; and second, at the time we conducted the experiments, BPEL related-tools proved to be not yet mature. Both points turned out to be a major hindrance for implementing preservation workflows by our targeted user group (preservation experts, archivists, data curators). Consequently, we chose to implement a much simplified, custom workflow description language and corresponding execution engine.

It is a crucial requirement of the programming language and execution environment that they allow the targeted community to assemble and deploy the preservation workflows they require, without forcing them to understand the underlying system. The WEE version 2 employs the concept of workflow templates which are pre-defined workflow definitions that can be easily instantiated and executed through a set of Web service interfaces.

2. Main Components and Services

2.1 Definitions

Workflow

A Planets preservation workflow defines a sequence of invocations of Planets Services, in which the output parameters of a given service are validly mapped to the input parameters of the following service. A Planets service is a software component that implements one of the specified Planets service interfaces (like Identify, Modify or Migrate). Workflow definitions contain activities (service invocations), as well as decision logic, loops, conditions and other control structures.

Workflow Template

A workflow template is a pre-defined workflow definition in which the nodes of the preservation sequence are service placeholders rather than concrete service endpoint. A service placeholder is a typed service which only defines the abstract interface while the actual implementation is undefined. The point of such a template is that a normal end user can select any service placeholder and replace it by a real service that implements the defined interface. This approach allows normal users to create Planets workflows without being forced to deal with the full complexity of the underlying system.

Workflow Configuration

A workflow configuration identifies a workflow template, the service endpoints associated with all template placeholders, and the parameters associated with each service. Hence, a Planets workflow is defined by a workflow configuration and its associated template implementation.

2.2 Workflow Execution Engine

The workflow execution engine (WEE) currently provides two main functionalities to a user:

- A web service for browsing the workflow template repository and
- A service for the execution and monitoring of workflow instances.

Workflow templates are pre-defined workflow fragments that must be completed by the user for example by specifying a particular Web service endpoint and/or service parameters. Workflow templates are being defined by experts and made available to Planets users through import into a Template Registry. The *WorkflowTemplateRegistry* provides service interfaces to register, browse, and retrieve workflow template definitions.

In order to schedule a workflow execution, a user submits a descriptor document as well as a pointer to the data registry to the *WorkflowExectuionManager*. A descriptor basically contains the identifier of a template and defines its parameterization (section 4). The main purpose of the WEE is the provision of a controlled environment for the specification and execution of preservation processes. Planets preservation workflows are built from Java components and serialized as XML workflow documents. For this purpose, the workflow engine implements an enactor that governs the orchestration of the various preservation components, which encapsulate functionalities like communication, state management, and preservation metadata handling. Preservation components can be implemented as local desktop components (e.g. for data model manipulations) or can abstract remote applications and services.

The Workflow Control Panel (WCP) complements the workflow environment by providing a generic portal client. The WCP provides a graphical user interface that allows one to choose from various abstract workflow scenarios (templates). The workflow templates are then rendered and visualized. Selected workflow templates are configured using the GUI representation (e.g. drop-down boxes) and can be executed and monitored using the Web application. Additionally, users can upload/generate an XML representation of the actual workflow instance.

From the user's point of view, the workflow execution environment (WEE) provides high-level interfaces for discovering, configuring, and instantiating preservation workflows. The WEE performs workflow execution asynchronously and may deliver status information to the user based on inquiry and email notification capabilities. It currently provides a generic web browser client or can be accessed by end user applications using web service or native interfaces. For future versions, we plan to incorporate a graphical workflow editor that supports our Java-based component model.

3. Utilizing the Workflow Environment

3.1 Downloading and Installation

Code Repository and Installer

The Planets workflow environment is contained within the interoperability framework source project. The related installation package and API can be downloaded from the Planets IF GForge project here: http://gforge.planets-project.eu/gf/project/if_sp/

The Planets software prototype code can be browsed though the Planets IF GForge project here: http://gforge.planets-project.eu/gf/project/if_sp/scmsvn/

Prerequisites

This guideline demonstrates the Planets workflow executing services based on a simple example. The required template implementation and xml configuration is packaged with IF release version 4. Prerequisites for installing and running the example are:

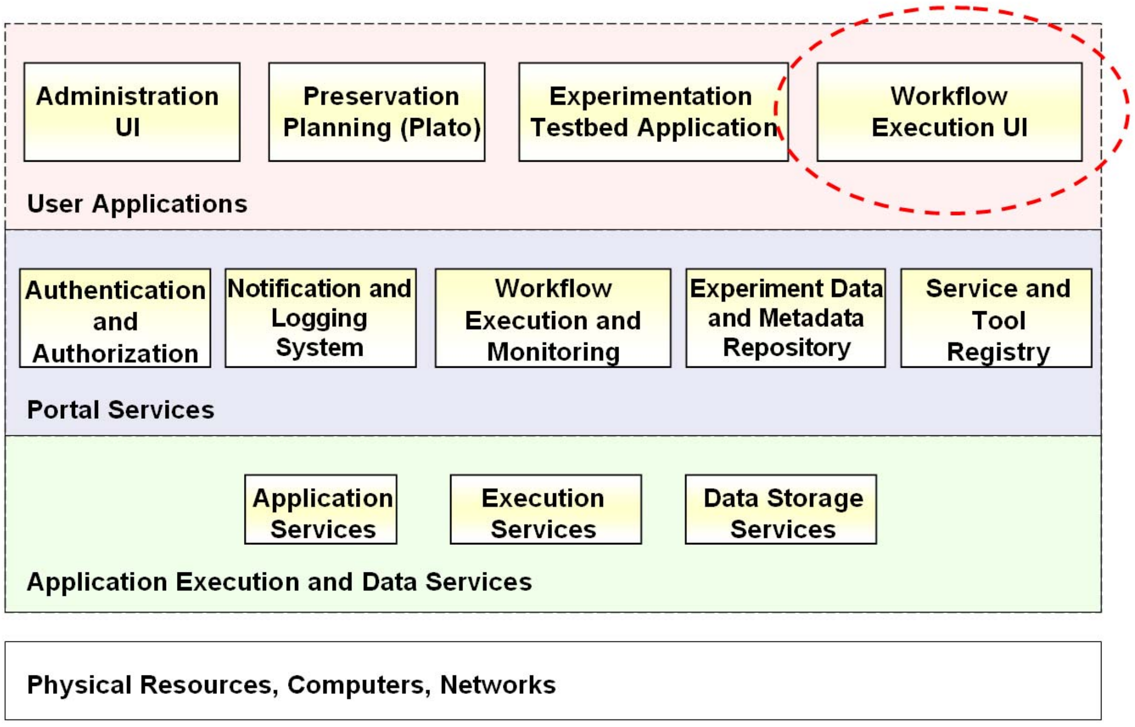
- The J2SE 1.5 Java Runtime Environment (JRE)
- A running installation of the Planets interoperability framework, version 4
    - Please consult the INSTALL_IF.txt file if built from source or
    - simply use the installer provided on the GForge page
- A set of installed and registered Planets services (section A.1)
    - required service for the example are the
        - DROID Identification Service
        - ImageMagickMigrate
        - SanselanIdentify Service
    - Pure Java services are packaged with the IF installer, for services that require additional software installed (e.g. ImageMagick) please consult the documentation of the Planets services project (Pserv).

For a detailed description on how to install the Planets Interoperability Framework and Preservation Services please refer to the latest release documentation.

For documentation on how to implement workflow templates please consult the developer and source documentation available at the Planets IF code repository.

3.2 The Workflow Control Panel (WCP)

The Planets workflow environment provides a set Web services (Workflow Execution and Monitoring) described later in this document. This Web service API allows Planets applications to remotely create, execute, and monitor complex preservation processes based on workflow documents. The Workflow Control Panel (WCP) provides a generic graphical user interface for the informal utilization of the workflow environment. Other applications that need to interface with the WEE are the Preservation Planning and the Testbed Application.


Administration UI
Preservation Planning (Plato)
Experimentation Testbed Application
Workflow Execution UI
User Applications
Authentication and Authorization
Notification and Logging System
Workflow Execution and Monitoring
Experiment Data and Metadata Repository
Service and Tool Registry
Portal Services
Application Services
Execution Services
Data Storage Services
Application Execution and Data Services
Physical Resources, Computers, Networks

Planets Service Architecture

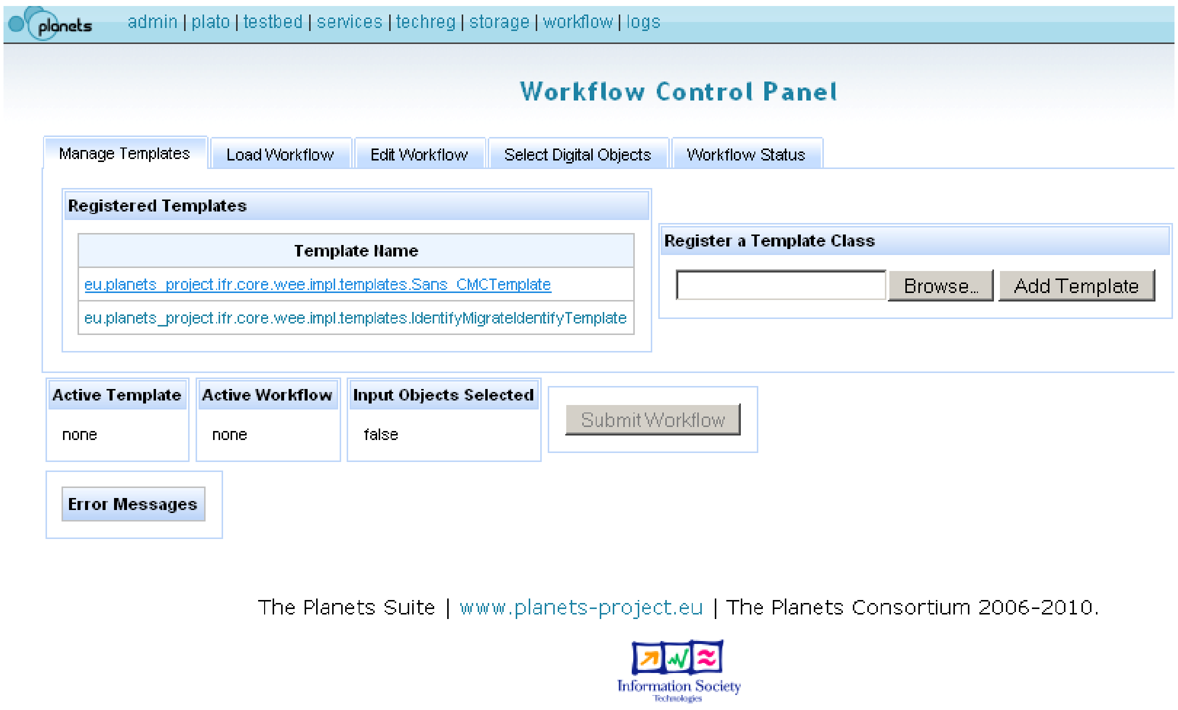
3.3 Editing and Executing Workflow Templates

In this section we provide a walkthrough for the Planets Workflow Control Panel based on a simple example workflow. The workflow first identifies the filetype of a *digital object,* migrates it to the Portable Network Graphics (PNG) format, and again runs another identification using a second identification tools. The example workflow template implementation as well as the configuration xml file is provided with the Planets source distribution.

There are five tabbed panes on the main WCP page, with the following functionalities:

1) Selecting Workflow Template

This tab provides a list of registered workflow templates, as well as an option for uploading new template source files (if authorized). Clicking on one of the registered templates will select that template as the **Active Template** (Note: this will erase any workflow previously held in memory) and direct the user to the Edit Workflow tab (the active workflow will be pre-configured for the selected template). The component does carry out a rough validation (it checks whether the provided template implements the correct interface) but does not pre-compile the template.


planets
admin | plato | testbed | services | techreg | storage | workflow | logs
Workflow Control Panel
Manage Templates
Load Workflow
Edit Workflow
Select Digital Objects
Workflow Status
Register a Template Class
Browse...
Add Template
Submit Workflow
Error Messages
The Planets Suite | www.planets-project.eu | The Planets Consortium 2006-2010.
Information Society Technologies

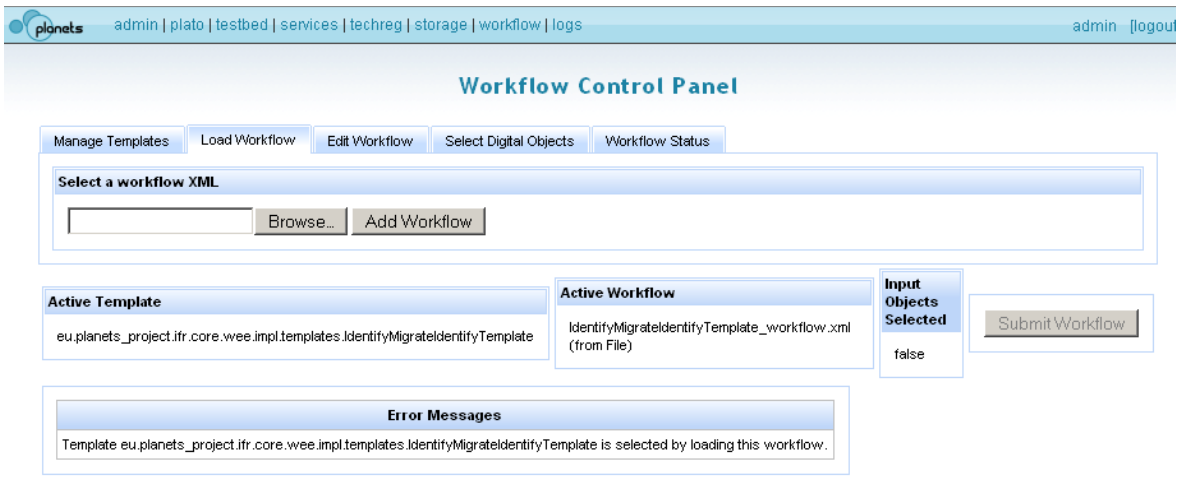
Registered Templates

Template Name
eu.planets_project.ifr.core.wee.impl.templates.Sans_CMCTemplate
eu.planets_project.ifr.core.wee.impl.templates.IdentifyMigrateIdentifyTemplate

Active Template	Active Workflow	Input Objects Selected
none	none	false

2) Loading a Workflow Configuration

This tab allows the user to upload an XML workflow description file (e.g. an executable preservation plan produced by the PLATO application). The workflow must validate against the planets_wtd.xsd schema in order to be successfully loaded. Furthermore, the template referenced by the XML workflow description <class> element must be loaded in the Template Registry. If these two conditions are met, the workflow will be successfully loaded, the referenced workflow template will be automatically set as the **Active Template** (this step will be reported in the Error Message box, although it is not an error per se), and the loaded workflow description will be set as the **Active Workflow**. Note that the **Active Workflow** will be tagged with the label "(from File)" as long as the loaded description remains unmodified. More detailed information on the workflow configuration schema is provided in section 4.



Workflow Control Panel

Select a workflow XML

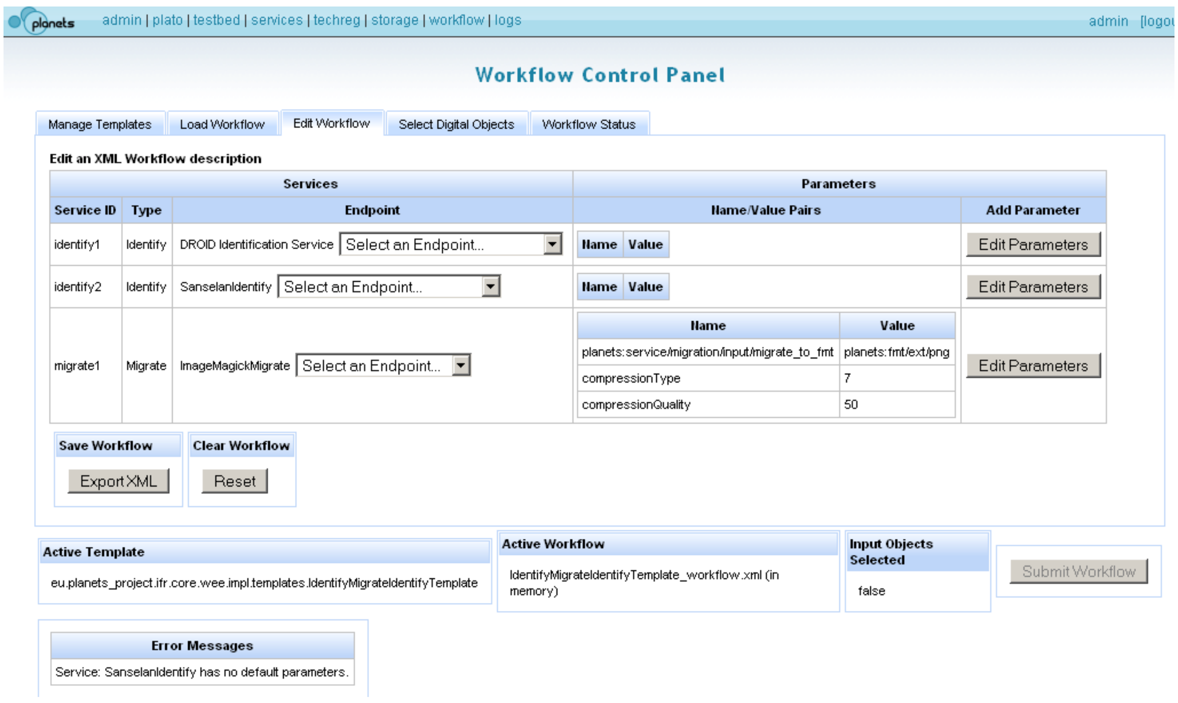
Active Template	Active Workflow	Input Objects Selected
eu.planets_project.ifr.core.wee.impl.templates.IdentifyMigrateIdentifyTemplate	IdentifyMigrateIdentifyTemplate_workflow.xml (from File)	false

Error Messages

Template eu.planets_project.ifr.core.wee.impl.templates.IdentifyMigrateIdentifyTemplate is selected by loading this workflow.

3) Editing Workflow and Exporting the Configuration File

In order to use this tab, a workflow template must be selected, either by selecting a template on the Manage Templates tab, or by loading a valid workflow description from the Load Workflow Tab. The user can then select service endpoints for each Service ID in the workflow from a drop-down list that is populated by a Service Registry query for all registered services of the defined Type. The user can also edit the name-value pairs that define the input parameters for a given service (the default parameters will be loaded from the Service Registry when an endpoint is selected) – refer to the **Workflow Parameters Panel** description below. Note that as soon as a modification to a loaded workflow is made, the label "(in memory)" will be attached to the Active Workflow name. The user can choose to save the present workflow description as an XML file to the local file system by clicking on the "Export XML" button. The user can also choose to clear the workflow (deselect all service endpoints, remove all parameters) by clicking on the "Reset" button.



Workflow Control Panel

Edit an XML Workflow description

Services			Parameters	
Service ID	Type	Endpoint	Name/Value Pairs	Add Parameter
identify1	Identify	DROID Identification Service [Select an Endpoint...]	Name / Value	Edit Parameters
identify2	Identify	SanselanIdentify [Select an Endpoint...]	Name / Value	Edit Parameters
migrate1	Migrate	ImageMagickMigrate [Select an Endpoint...]	planets:service/migration/input/migrate_to_fmt = planets:fmt/ext/png; compressionType = 7; compressionQuality = 50	Edit Parameters

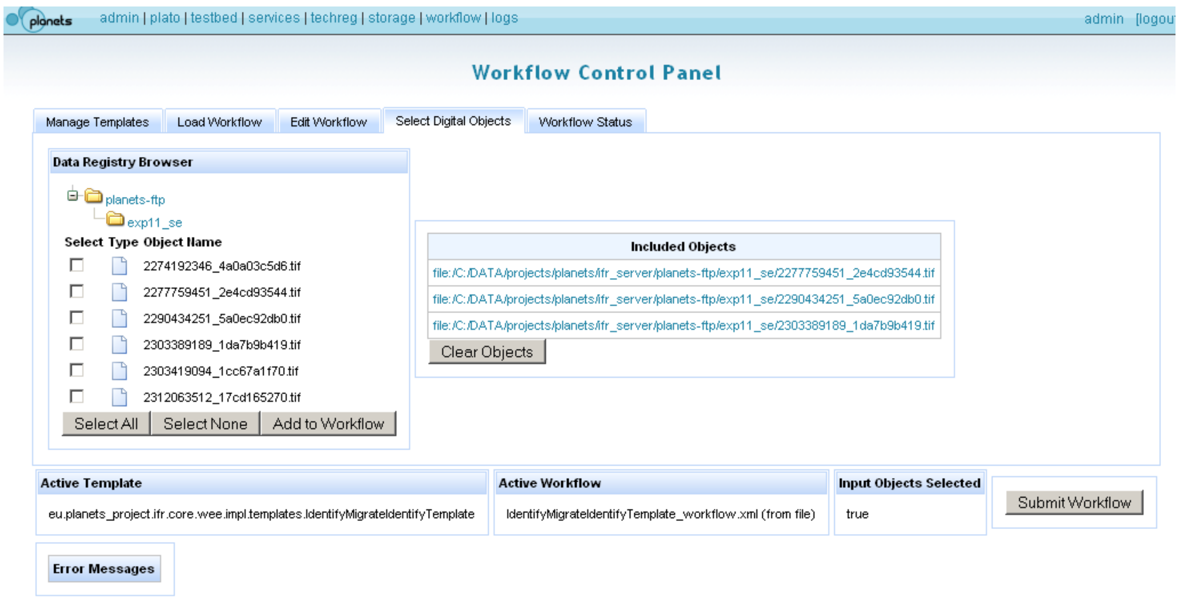
Active Template	Active Workflow	Input Objects Selected
eu.planets_project.ifr.core.wee.impl.templates.IdentifyMigrateIdentifyTemplate	IdentifyMigrateIdentifyTemplate_workflow.xml (in memory)	false

Error Messages

Service: SanselanIdentify has no default parameters.

4) Selecting Digital Objects to Process

Here the user can browse a hierarchical display of data sources; presently a file-system data source and a Data Manager for the Planets Amazon Simple Storage Service (S3) account are pre-configured. The user can choose to add selected Digital Objects to the Included Objects list, or can clear this list. If al required prerequisites are met, the user can submit a workflow to the job queue.



Select	Type	Object Name
☐		2274192346_4a0a03c5d6.tif
☐		2277759451_2e4cd93544.tif
☐		2290434251_5a0ec92db0.tif
☐		2303389189_1da7b9b419.tif
☐		2303419094_1cc67a1f70.tif
☐		2312063512_17cd165270.tif

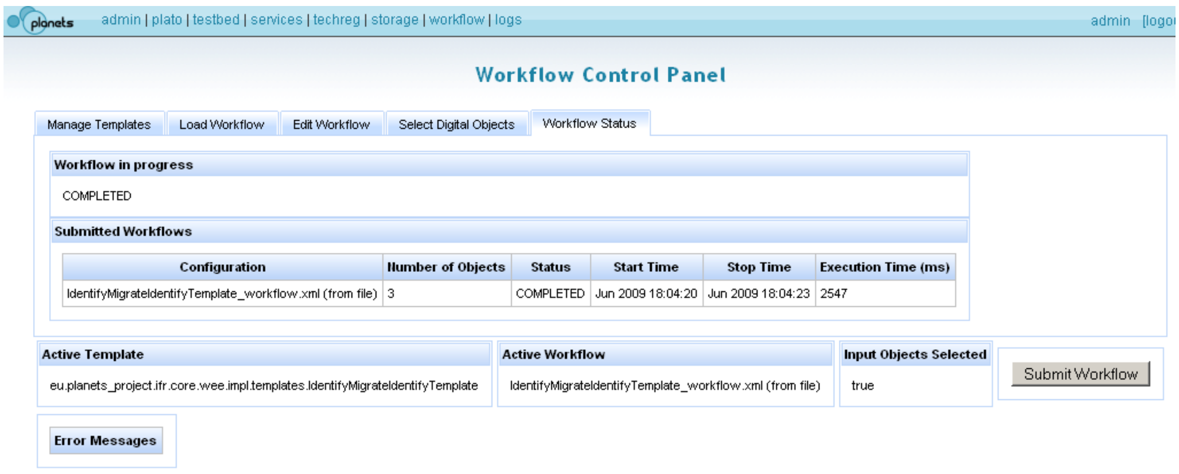
Included Objects
file:/C:/DATA/projects/planets/ifr_server/planets-ftp/exp11_se/2277759451_2e4cd93544.tif
file:/C:/DATA/projects/planets/ifr_server/planets-ftp/exp11_se/2290434251_5a0ec92db0.tif
file:/C:/DATA/projects/planets/ifr_server/planets-ftp/exp11_se/2303389189_1da7b9b419.tif

Active Template	Active Workflow	Input Objects Selected
eu.planets_project.ifr.core.wee.impl.templates.IdentifyMigrateIdentifyTemplate	IdentifyMigrateIdentifyTemplate_workflow.xml (from file)	true

5) Checking the Workflow Execution Status

A workflow can be submitted when an **Active Template** is defined, when an **Active Workflow** is defined, and when **Input Objects Selected** is true. When a workflow is submitted, the user is automatically directed to the Workflow Status tab.

This tab presents a table of information about workflows that have been submitted. At present, this information is not persistent; that is, only information about workflows submitted during the current user session is displayed.



Workflow in progress

COMPLETED

Submitted Workflows

Configuration	Number of Objects	Status	Start Time	Stop Time	Execution Time (ms)
IdentifyMigrateIdentifyTemplate_workflow.xml (from file)	3	COMPLETED	Jun 2009 18:04:20	Jun 2009 18:04:23	2547

Active Template	Active Workflow	Input Objects Selected
eu.planets_project.ifr.core.wee.impl.templates.IdentifyMigrateIdentifyTemplate	IdentifyMigrateIdentifyTemplate_workflow.xml (from file)	true

Workflow Parameters Panel

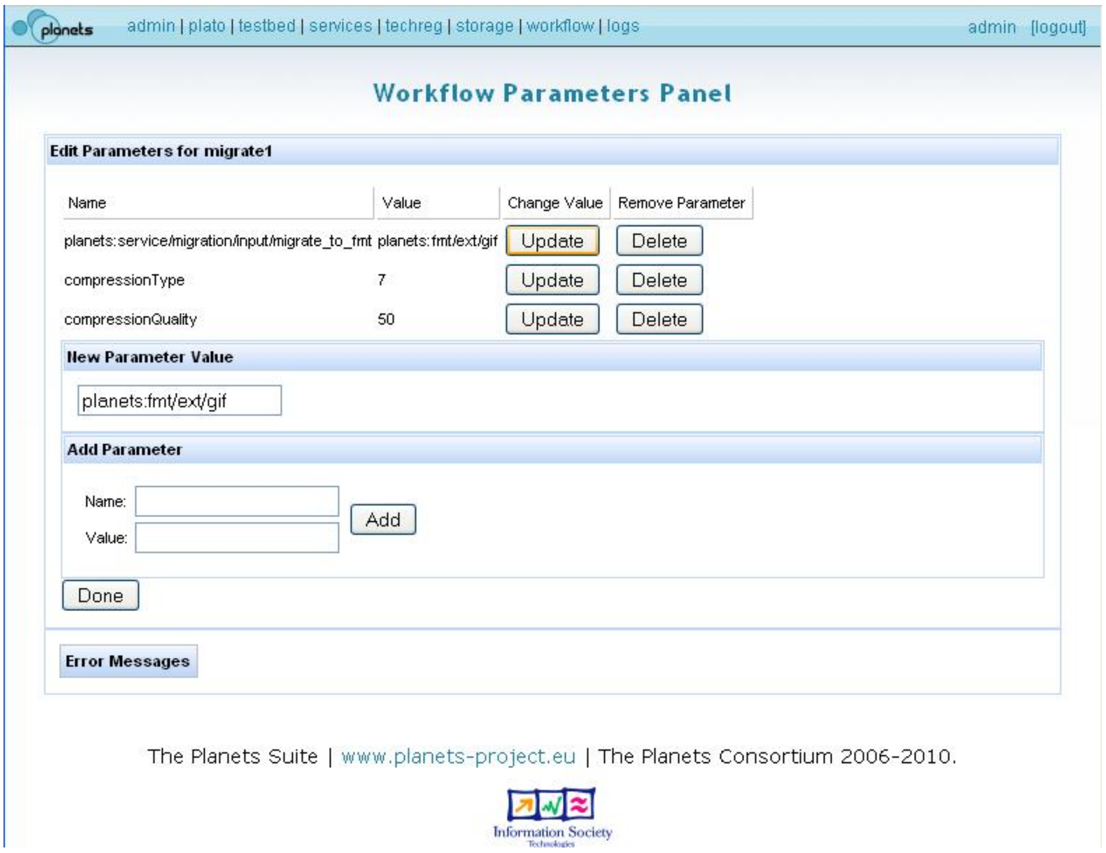
A list of parameters associated with a given service placeholder are shown in a popup window titled “Workflow Parameters Panel”. The header of the main panel has the title “Edit Parameters for <name>”, where <name> is the service placeholder (or Service ID) in question.

If the user wishes to remove an existing parameter, she should click the “Delete” button associated with that parameter.

If the user wishes to update an existing parameter, she should enter the new parameter value in the “New Parameter Value” text field, and then click on the “Update” button associated with the desired parameter.

If the user wishes to define a new parameter, she should enter a Name and Value in the “New Parameter” text fields and then click on the “Add” button.

When the user is finished modifying the parameters, she should click on the “Done” button; the window will close and the focus will return to the original Edit Workflow panel.


planets
admin | plato | testbed | services | techreg | storage | workflow | logs
admin [logout]

Workflow Parameters Panel

Edit Parameters for migrate1

Name	Value	Change Value	Remove Parameter
planets:service/migration/input/migrate_to_fmt	planets:fmt/ext/gif	Update	Delete
compressionType	7	Update	Delete
compressionQuality	50	Update	Delete

New Parameter Value

planets:fmt/ext/gif

Add Parameter

Name:
Value:

The Planets Suite | www.planets-project.eu | The Planets Consortium 2006-2010.

4. Workflow Configuration Schema

This section describes the basic elements of the workflow configuration schema. The schema defines the XML representation of the particular workflow configuration which is used to instantiate a workflow template. Workflows configurations can be created and serialized (as XML files) using the WCP or by any other tool that supports the configuration schema. The latest release of the schema definition can be found at the Planets source repository (planets_wdt.xsd).

The figure below demonstrates a simple workflow configuration specifying a workflow template, a set of services, as well as service parameters. For more detailed information the reader is referred to the schema definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<workflowConf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="planets_wdt.xsd">

<template>
  <!-- fully qualified name of template class -->
  <class>eu.planets.project.ifr.core.wee.impl.templates.Sans_CMCTemplate</class>
</template>
<!-- list of services -->
<services>
  <!-- defines service id and endpoint -->
  <service id="identify1">
    <endpoint>http://planets1.arcs.ac.at/pserv-pc-droid/Droid?wsdl</endpoint>
  </service>
  <service id="identify2">
    <endpoint>http://planets1.arcs.ac.at/pserv-pa-sanselan/Sanselan?wsdl</endpoint>
  </service>
  <service id="migrate1">
    <endpoint>http://planets1.arcs.ac.at/pserv-pa-imagemagick/ImageMagick?wsdl</endpoint>
    <!-- optional list of service parameters -->
    <!-- defined in registered service description document -->
    <parameters>
      <param>
        <name>planets:service/migration/input/migrate_to_fmt</name>
          <value>planets:fmt/ext/png</value>
        </param>
        <param>
          <name>compressionType</name>
          <value>7</value>
        </param>
        <param>
          <name>compressionQuality</name>
          <value>50</value>
        </param>
      </parameters>
    </service>
  </services>
</workflowConf>
```

5. Implementation Details and Public Interfaces

The main elements of the Workflow Execution Engine package are

- Two web-services / stateless-session-beans which provide the application's interface
    - **WorkflowTemplateRegistry**: contains methods for browsing, registering and retrieving workflow Templates
    - **WorkflowExecutionManager**: contains methods for submitting a workflow and polling for the execution's status and result
- A message-driven-bean: the WorkflowExecutionEngine - the job-processor

WorkflowTemplateRegistry

The WorkflowTemplateRegistry provides an implementation for browsing, retrieving and submitting WorkflowTemplates. The following methods are provided:

- `getAllSupportedQNames`: Returns a list of all QNames of registered WFTemplates
- `getWFTemplate(String QWorkflowTemplateName` : Returns the source (Java Workflow Template .java file) for a registered template. This might be necessary to understand a template's behaviour.
- `registerWorkflowTemplate(byte[] javaBinary)`: Registers a new WorkflowTemplate

The parameter `QWorkflowTemplateName` is the fully qualified name of the submitted class. e.g. '`eu.planet_project.ifr.core.Template1.java`'. The parameter `javaBinary` is a byte array of the java class. The template will be added to the template registry and persisted using the Planets data registry.

WorkflowExecutionManager

The WorkflowExecutionManager is a web service / stateless-session bean interface that offers the following method:

- `public UUID submitWorkflow(List digitalObjects, String QWorkflowTemplateName, String xmlWorkflowConfig)`

`DigitalObjects` is a list of Planets Digital Objects, which provide the payload a workflow is invoked upon; `QWorkflowTemplateName` the fully qualified name of the java workflowTemplate to use for this submission. i.e. specifying the structure. (Please note: the template must be registered!); `xmlWorkflowConfig` is an XML String that defines the workflowTemplate's configuration i.e. specifying how to populate a static template. The return value is a ticket (UUID) for the submitted job which can be used for polling information about the workflow:

- querying on a job's status and its position in queue
- retrieving execution results

WorkfowFactory

The WorkflowFactory class contains a single public Method:

- `public static WorkflowInstance create(WorkflowConf wfConf, List<DigitalObject> digos) throws Exception`

The class implements a builder that

- extracts the information handed over in the xml workflow configuration (workflowTemplate's QName to use, serviceIDs, endpoints, their parameters, etc.)
- queries the registry and fetches the requested workflowTemplate.java, followed by dynamically validating, compiling, jar-ing and classloading the required workflowTemplate
- reflects on the template's fields and
    - o builds and initializes proxies for the passed service endpoints and stores the service parameter configuration
    - o includes the data to invoke the workflow upon (in terms of DigitalObjects)
    - o and returns a WorkflowInstance Object which can be submitted to the queue and executed upon the WorkflowExecutionEngine.

WorkflowTemplate

A workflowTemplate is a java class that defines

- the service-interface-types which are used (e.g. Identification or Migration Interface) but not the actual service-instances (e.g. Droid or ImageMagick)
- the structure and process flow within a workflow (e.g. branching, looping, exception handling, flow of level-one-Service input and output data, etc.)
- results within the data registry and documents all actions in the `workflowResult` object which is returned

A workflow template may contain any service that implements the Planets Service Interface. In order to provide a valid implementation one needs to be aware of the following conventions:

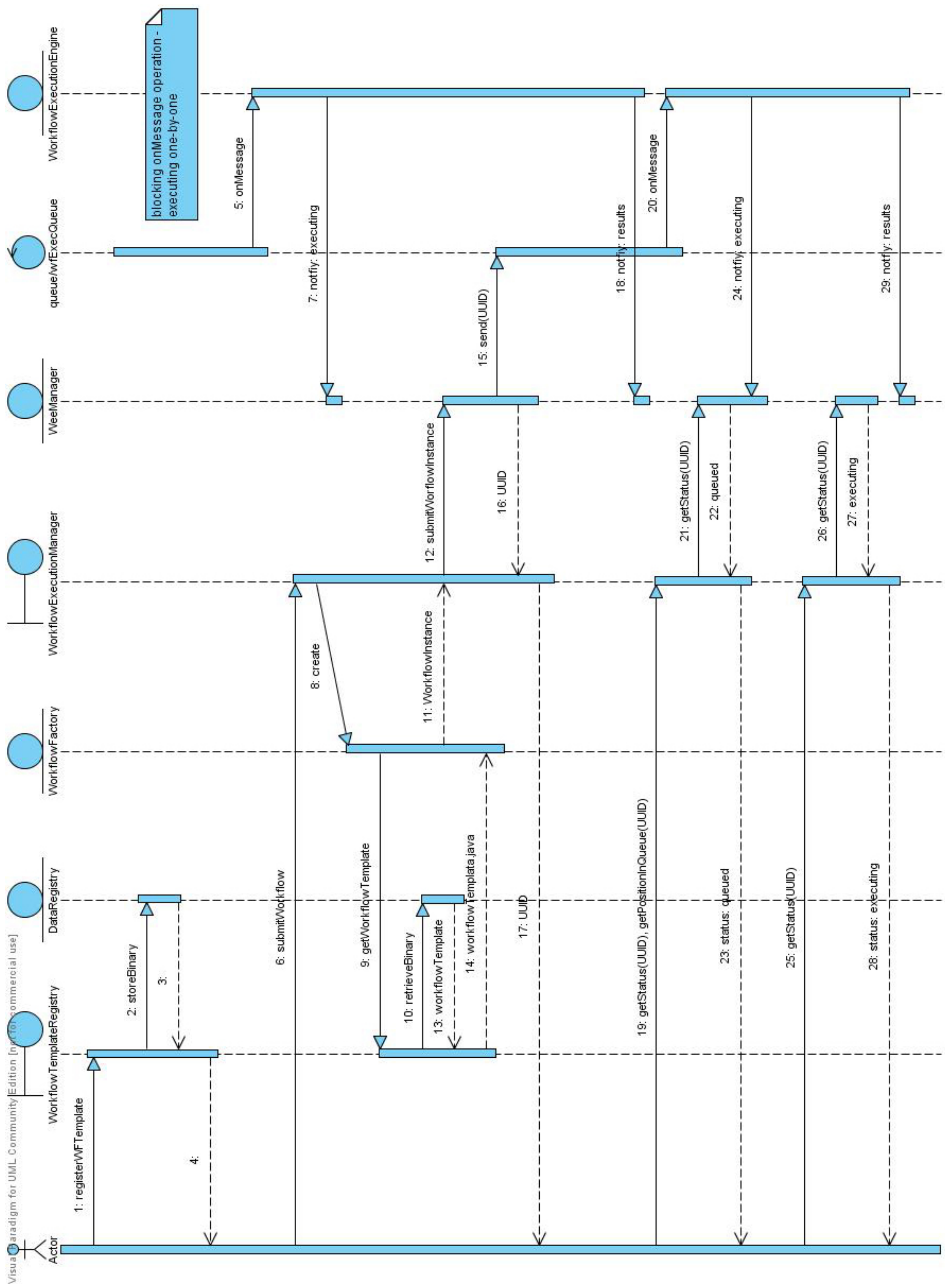
- Naming convention: The id used within the workflowconfig.xml must corresponds to the object's variable name within the template class, e.g.
  `<service id="identify1">  -> private Identify identify1;`
- The WorkflowFactory validates if the ServiceTemplate interface is implemented.
- The WorkflowFactory checks for all defined Service interfaces which are used to construct the business logic of a workflow if they are in the range of supported PlanetsServiceTypes

Currently supported service types are:

- `eu.planets_project.services.identify.Identify`
- `eu.planets_project.services.characterise.Characterise`
- `eu.planets_project.services.characterise.DetermineProperties`
- `eu.planets_project.services.compare.BasicCompareFormatProperties`
- `eu.planets_project.services.migrate.Migrate`
- `eu.planets_project.services.migrate.MigrateAsync"`

Planets Preservation Workflow Sequence Diagram

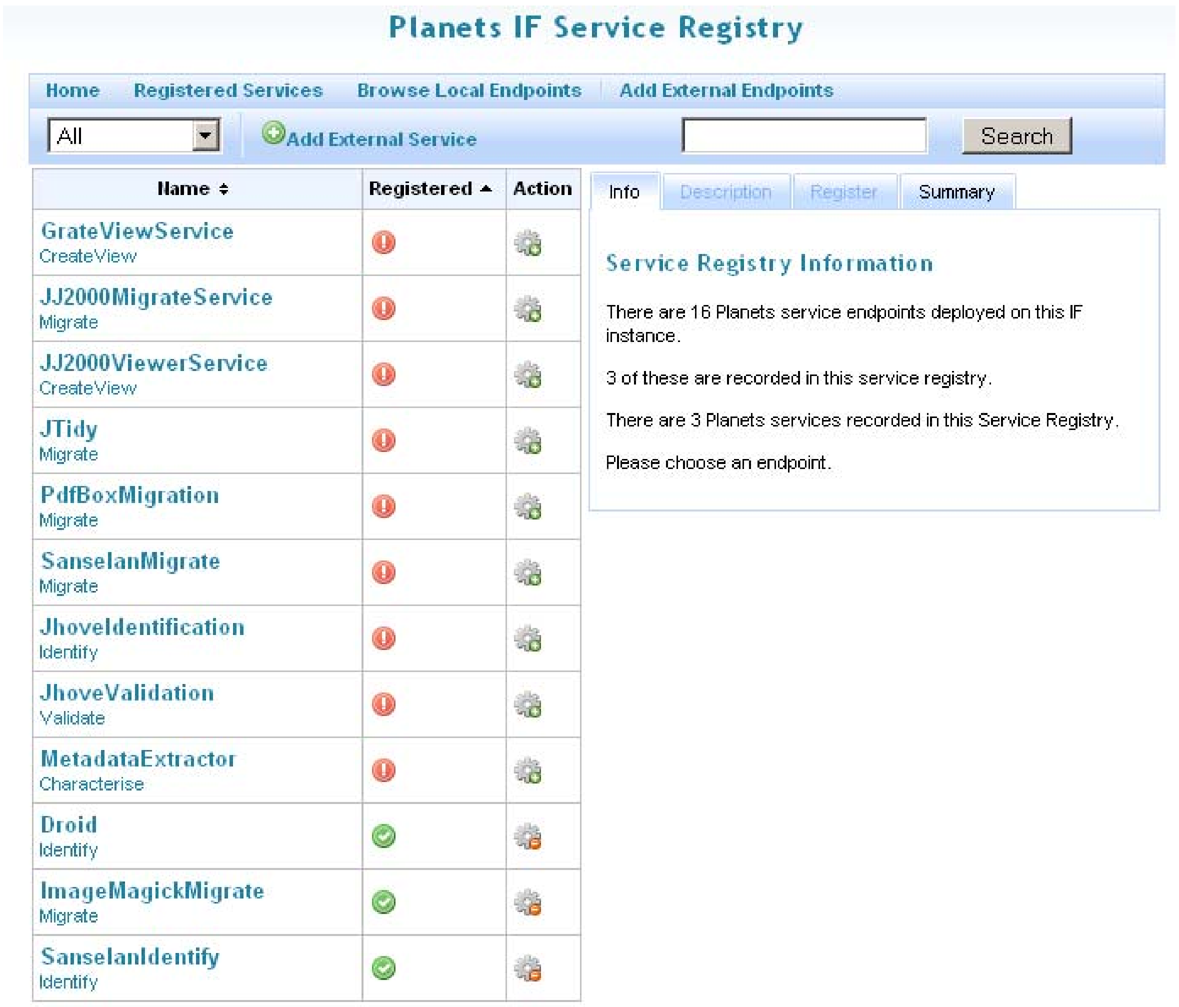
A short example using a sequence diagram on registering a workflowTemplate and on running a Workflow Execution is shown in the figure below:



Workflow Execution Sequence Diagram

A.1 Service Registry Browser

The Service Registry Browser provides a user interface for registering local and remote service endpoints with the service registry, which generates an instance of a rich service descriptor for each preservation service endpoint. This information is generated once the service is registered with the preservation system and needs to be supported by a service description port type.



Name	Registered	Action
GrateViewService CreateView	●	⚙
JJ2000MigrateService Migrate	●	⚙
JJ2000ViewerService CreateView	●	⚙
JTidy Migrate	●	⚙
PdfBoxMigration Migrate	●	⚙
SanselanMigrate Migrate	●	⚙
JhoveIdentification Identify	●	⚙
JhoveValidation Validate	●	⚙
MetadataExtractor Characterise	●	⚙
Droid Identify	✓	⚙
ImageMagickMigrate Migrate	✓	⚙
SanselanIdentify Identify	✓	⚙

Service Registry Information

There are 16 Planets service endpoints deployed on this IF instance.

3 of these are recorded in this service registry.

There are 3 Planets services recorded in this Service Registry.

Please choose an endpoint.