Project no. 034567

Grid4All

Specific Targeted Research Project (STREP)

Thematic Priority 2: Information Society Technologies

Deliverable 3.5 – Implementation of Semantic Store layered above VOFS

Due date of deliverable: June, 2009

Actual submission date: July, 2009

Start date of project: 1 June 2006 Duration: 36 months

Organisation name of lead contractor for this deliverable: INRIA Regal

Contributors: Jean-Michel Busca

Editors: Marc Shapiro Release: 2009-07-15

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

1. Overview

The Telex middleware is managed under the GForge collaborative development environment. The Telex project is publicly available at http://telex2.gforge.inria.fr. Releases of Telex can be found under the “Files” tab of the project (http://gforge.inria.fr/frs/?group_id=750). Each contains the Telex library itself (jar file), the corresponding source code tree and the related documentation (papers, API javadoc, etc.). Three trackers allow users to report bugs, submit features requests and ask for support of the development team.

2. Functionality

The current release of Telex is V0.5. It implements the functionality presented in deliverable 3.1 “Requirement analysis, design and implementation plan of Grid4All data storage and sharing facility” and described in detail in deliverable 3.4 “Design and initial prototype of the Semantic Store”.

This release was used to develop and test the Shared Calendar and the Collaborative File Sharing applications of WP4,as well as a collaborative ontology editor at the university of the Aegean. As described in previous deliverables, this release of Telex can operate stand-alone or interface with VOFS for enhanced user experience.

3. Installation

Telex is a library: it must be deployed and installed as part of the deployment and installation process of the applications that use it. Telex operation can be configured through parameters stored in the “.telex.properties” configuration file, under the user's home directory. If this file does exist, Telex assumes default values for the parameters, which are suitable for production. The complete list of configuration parameters is documented in each release of Telex in the sample telex.properties file.

4. Telex API

The following pages present the javadoc of the API V0.5 generated from the source code.

Package
fr.inria.gforge.telex

Provides classes and interfaces of objects instanciated by Telex.

fr.inria.gforge.telex
Class ClosedDocumentException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-fr.inria.gforge.telex.ClosedDocumentException
```

All Implemented Interfaces:
java.io.Serializable

public class **ClosedDocumentException**
extends java.lang.Exception

Thrown when an application accesses a document that has been closed. The pathname of the document is provided in the detail message.
Author:
J-M. Busca INRIA/Regal

Field Summary

`public static final`	`serialVersionUID` Value: **1**

Constructor Summary

`public`	`ClosedDocumentException(java.lang.String message)`

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

serialVersionUID

`public static final long` **`serialVersionUID`**

Constant value: **1**

Constructors

(continued on next page)

(continued from last page)

ClosedDocumentException

```
public ClosedDocumentException(java.lang.String message)
```

fr.inria.gforge.telex
Interface Document

public interface **Document**
extends

A shared document edited by a `TelexApplication`. Each document is identified by an id of type `java.lang.String`, which is unique across all sites. This id is the absolute pathname of the document in some globally shared namespace. Each site maps the root of this shared namespace to the local directory of its choice.

This interface provides methods for viewing, updating and saving the document. These methods operate on the persistent state of the document, which comprises:

- A multi-log. It is the core data structure that represents the current state of the document. It contains a set of `Action`s, bound by `Constraint`s, that users submit to update to the document.
- A set of user-defined filters. Each user may define its own view of the document, by applying `ActionFilter`s. Filters are named and stored as part of the document state, in a per-user namespace.
- A set of per-user snapshots. Each user may define `StateSnapshot` of the document that are of particular interest to him. Snapshots are named and stored stored as part of the document state, in a per-user namespace.

When the document is open in `Telex.OpenMode.READ_ONLY` mode, only the methods that do not modify the persistent state of the document are allowed. When it is open in `Telex.OpenMode.READ_WRITE` or `Telex.OpenMode.CREATE` mode, all of the methods are allowed. Most of these methods operate on behalf of the user who invoked the Virtual Machine, hereafter referred to as the *invoking user*.

Document updating. Invoking user may update the document by adding actions and constraints through the `addAction(Action)`, `addConstraint(Constraint)` and `addFragment(Fragment)` methods. Telex propagates these actions and constraints to peer Telex sites when connectivity permits, using a best-effort epidemic replication protocol. Updates to the document are periodically notified to the application through the `TelexApplication.execute(Document, ScheduleGenerator)` method.

Document viewing. Invoking user may assign a set of filters to the document through the `defineFilter(ActionFilter)` and the `removeFilter(ActionFilter)` methods. Filters are saved as part of the persistent state of the document, in the name space of invoking user. When invoking user opens a document, the set of filters that he has defined for this document is automatically restored and applied. This set may be obtained through the `listFilters()` method.

Document saving. Invoking user may define a set of document snapshots to retain through the `defineSnapshot(StateSnapshot)` and the `removeSnapshot(StateSnapshot)` methods. Snapshots are saved as part of the persistent state of the document, in the name space of invoking user. The set of snapshots currently defined by invoking user may be obtained through the `listSnapshots()` method.

Author:
J-M. Busca INRIA/Regal

Field Summary

`public static final`	`DEFAULT_TYPE` The type of a document whose name has no suffix. Value:

Method Summary

`void`	`addAction(Action action)` Adds the specified action to this document.

void	`addConstraint(Constraint constraint)` Adds the specified constraint to this document.
void	`addFragment(Fragment fragment)` Adds the specified multi-log fragment to this document.
void	`addOverlappingFragment(Fragment fragment)` Adds the specified overlapping fragment to the (relevant) documents this document is bound to.
void	`close()` Closes this document.
boolean	`defineFilter(ActionFilter filter)` Adds the specified action filter to the set defined by invoking user for this document.
boolean	`defineSnapshot(StateSnapshot snapshot)` Adds the specified snapshot to the set defined by invoking user for this document.
boolean	`executeNow(boolean force)` Calls the `execute()` method on this document according to the specified mode.
void	`garbageCollect(StateSnapshot snapshot)` Garbage-collects the history of this document up to the specified snapshot.
TelexApplication	`getApplication()` Returns the application that is currently editing this document.
java.lang.String	`getId()` Returns the unique id of this document.
java.lang.String	`getPathname()` Returns the pathname of this document.
java.lang.String	`getType()` Returns the type of this document.
boolean	`isClosed()` Returns the open/closed status of this document.
boolean	`isOffline()` Returns the on-line/off-line status of this document.
ActionFilter[]	`listFilters()` Returns the set of action filters that are currently defined by invoking user for this document.
StateSnapshot[]	`listSnapshots()` Returns the set of snapshots that are currently defined by invoking user for this document.
boolean	`propose(Schedule decision)` Proposes the specified decisions (set of constraints).
boolean	`removeFilter(ActionFilter filter)` Removes the specified action filter from the set defined by invoking user for this document.
boolean	`removeSnapshot(StateSnapshot snapshot)` Removes the specified snapshot from the set defined by invoking user for this document.

void	`setSchedulingParameters(SchedulingParameters parameters)` Applies the specified scheduling parameters to this document.
void	`startFrom(StateSnapshot snapshot)` Requests Telex to start from the specified snapshot whenever possible when generating new schedules.

Fields

DEFAULT_TYPE

`public static final java.lang.String DEFAULT_TYPE`

The type of a document whose name has no suffix.
Constant value:

Methods

getId

`public java.lang.String getId()`

Returns the unique id of this document. This id is unique across all participating sites.

Returns:
the unique id of this document.

getPathname

`public java.lang.String getPathname()`

Returns the pathname of this document. This pathname is specific to the local site.

Returns:
the pathname of this document.

getType

`public java.lang.String getType()`

Returns the type of this document. The type of a document is the suffix of its name, or `DEFAULT_TYPE` if the name has no suffix.

Returns:
the string representing the type of this document, possibly DEFAUL_TYPE.

getApplication

`public TelexApplication getApplication()`

Returns the application that is currently editing this document.

Returns:
the the application that is currently editing this document.

(continued on next page)

(continued from last page)

isOffline

`public boolean **isOffline**()`

Returns the on-line/off-line status of this document.

Returns:
true if this document is closed or off-line, and false otherwise.

isClosed

`public boolean **isClosed**()`

Returns the open/closed status of this document.

Returns:
true if this document is closed, and false otherwise.

close

```
public void close()
  throws java.io.IOException
```

Closes this document. After this call returns, the application will not be allowed any action on this document, and it will stop being notified of new events regarding this document.

Throws:
`IOException` - if an I/O error occurs.

addAction

```
public void addAction(Action action)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidFragmentException,
         java.io.IOException,
         UnsoundGraphException
```

Adds the specified action to this document. This is a convenience method which is semantically equivalent to:

```
Fragment fragment = new Fragment();
fragment.add(action);
addFragment(fragment);
```

See `addFragment(Fragment)` for more details.

Parameters:
`action` - the action to add.

Throws:
`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidFragmentException` - if the specified action is invalid.
`IOException` - if an I/O error occurs.
`UnsoundGraphException` - if there is a consistency problem.

addConstraint

```
public void addConstraint(Constraint constraint)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidFragmentException,
         java.io.IOException,
         UnsoundGraphException
```

Adds the specified constraint to this document. This is a convenience method which is semantically equivalent to:

```
Fragment fragment = new Fragment();
fragment.add(constraint);
addFragment(fragment);
```

See `addFragment(Fragment)` for more details.

Parameters:

`constraint` - the constraint to add.

Throws:

`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidFragmentException` - if the specified constraint is invalid.
`IOException` - if an I/O error occurs.
`UnsoundGraphException` - if there is a consistency problem.

addFragment

```
public void addFragment(Fragment fragment)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidFragmentException,
         java.io.IOException,
         UnsoundGraphException
```

Adds the specified multi-log fragment to this document. The specified fragment must be valid: (i) none of its action must have been written to a document yet, (ii) all of its constraints must bind actions that are either in the fragment, or already written to a document. This document must be open in read-write mode. Depending on the current `SchedulingParameters` values, a call to this method this will trigger a call to the `TelexApplication.execute(Document, ScheduleGenerator)` method.

Telex provides the following guarantees:

- Telex will associate the fragment's actions and constraints to this document before calling the application's constraint checker. Action.getDocument(), Action.getPeer(), etc. methods will thus return valid values when called within the constraint checker.
- If the threshold of fragments is reached when calling this method, the execute() method will be called within the current thread. That is, the execute() method will complete before this method returns.
- All actions and constraints submitted within the specified fragment will be propagated to other peers and added to their action-constraint graph atomically.

Parameters:

`fragment` - the fragment to add.

Throws:

(continued on next page)

(continued from last page)

`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidFragmentException` - if the specified fragment is invalid.
`IOException` - if an I/O error occurs.
`UnsoundGraphException` - if there is a consistency problem

addOverlappingFragment

```
public void addOverlappingFragment(Fragment fragment)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidFragmentException,
         java.io.IOException,
         UnsoundGraphException
```

Adds the specified overlapping fragment to the (relevant) documents this document is bound to. A fragment is overlapping if it contains actions that must be written to distinct (bound) documents. The document each action of the fragment must be written to (or target document) must be specified prior to calling this method through `Action.setDocument(Document)` . This method writes each action in the target document, together with all intra-document constraints. It then writes cross-document constraints in relevant documents. (A constraint set between documents d1 and d2 is stored both in d1 and d2.)

The specified fragment must be valid: (i) none of its action must have been written to a document yet, (ii) all of its constraints must bind actions that are either in the fragment, or already written to a document, (iii) all target documents must be bound to this document. Target documents must be open in read-write mode. Depending on the current `SchedulingParameters` values, a call to this method this will trigger a call to the `TelexApplication.execute(Document, ScheduleGenerator)` method.

Telex provides the following guarantees:

- Telex will associate the fragment's actions and constraints to this document before calling the application's constraint checker. Action.getDocument(), Action.getPeer(), etc. methods will thus return valid values when called within the constraint checker.
- If the threshold of fragment is reached when calling this method, the execute() method will be called within the current thread. That is, the execute() method will complete before this method returns.
- All actions and constraints submitted within the specified fragment will be propagated to other peers and added to their action-constraint graph atomically on a **per-document** basis.

Parameters:
`fragment` - the fragment to add.

Throws:
`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidFragmentException` - if the specified fragment is invalid.
`IOException` - if an I/O error occurs.
`UnsoundGraphException` - if there is a consistency problem

executeNow

```
public boolean executeNow(boolean force)
```

Calls the `execute()` method on this document according to the specified mode. The method checks whether (i) actions or constraints have been added to the document or (ii) the set of active filters has changed since the last call to the execute() method. If none of these conditions is true, the method does not call the execute() method unless the force parameter is set to true. In the latter case, the method does not wait for the application to release the schedule generator: two or more threads may then execute the execute() method concurrently. If the execute() method is actually called, it executes within the current (invoking) thread, and completes before this method returns.

Parameters:
`force` - the execute() method is called unconditionally when true, or only if necessary when false.

(continued on next page)

(continued from last page)

Returns:
true if the execute() has been called, and false otherwise.

startFrom

```
public void startFrom(StateSnapshot snapshot)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidScheduleException
```

Requests Telex to start from the specified snapshot whenever possible when generating new schedules. The request takes effect in the next call to the `execute()` method after this method returns.

Parameters:
`snapshot` - the snapshot to start from.

Throws:
`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidScheduleException` - if the specified schedule does not relate to this document.

propose

```
public boolean propose(Schedule decision)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidScheduleException,
         java.io.IOException
```

Proposes the specified decisions (set of constraints).

Parameters:
`decision` - a schedule to vote for.

Returns:
true if the schedule was successfully proposed

Throws:
`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidScheduleException` - if the specified schedule is not legal.
`IOException` - if an I/O error occurs.
`ReconciliationException` - (to be completed.)

setSchedulingParameters

```
public void setSchedulingParameters(SchedulingParameters parameters)
```

Applies the specified scheduling parameters to this document.

Parameters:
`parameters` - the scheduling parameters to apply.

defineFilter

```
public boolean defineFilter(ActionFilter filter)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         java.io.IOException
```

(continued on next page)

(continued from last page)

Adds the specified action filter to the set defined by invoking user for this document. If this method returns successfully, the specified filter is permanently saved as part of the persistent state of this document. This method itself does not force immediate re-generation of sound schedules: call the `executeNow(boolean)` method to do so.

Parameters:

`filter` - the filter to define.

Returns:

true if the filter is actually added, and false if the filter already existed for invoking user.

Throws:

`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`IOException` - if an I/O error occurs.

removeFilter

```
public boolean removeFilter(ActionFilter filter)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         java.io.IOException
```

Removes the specified action filter from the set defined by invoking user for this document. If the method returns successfully, the specified filter is permanently removed from the persistent state of this document. This method itself does not force immediate regeneration of sound schedules: call the `executeNow(boolean)` method to do so.

Parameters:

`filter` - the filter to remove.

Returns:

true if the filter is actually removed, and false if the filter did not exist for invoking user.

Throws:

`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`IOException` - if an I/O error occurs.

listFilters

```
public ActionFilter[] listFilters()
  throws ClosedDocumentException,
         java.io.IOException
```

Returns the set of action filters that are currently defined by invoking user for this document.

Returns:

the current set of action filters.

Throws:

`ClosedDocumentException` - if this document is closed.
`IOException` - if an I/O error occurs.

defineSnapshot

```
public boolean defineSnapshot(StateSnapshot snapshot)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidSnapshotException,
         java.io.IOException
```

(continued on next page)

(continued from last page)

Adds the specified snapshot to the set defined by invoking user for this document. If this method returns successfully, the specified snapshot is permanently saved as part of the persistent state of this document.

Parameters:

`snapshot` - the snapshot to define.

Returns:

true if the snapshot is actually saved under its name, and false if a snapshot with the same name already existed for invoking user.

Throws:

`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidSnapshotException` - if the specified snapshot does not relate to this document.
`IOException` - if an I/O error occurs.

removeSnapshot

```
public boolean removeSnapshot(StateSnapshot snapshot)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         InvalidSnapshotException,
         java.io.IOException
```

Removes the specified snapshot from the set defined by invoking user for this document. If the method returns successfully, the specified snapshot is permanently deleted from the persistent state of this document.

Parameters:

`snapshot` - the snapshot to remove.

Returns:

true if the snapshot is actually removed, and false if the snapshot was not saved for invoking user.

Throws:

`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`InvalidSnapshotException` - if the specified snapshot does not relate to this document.
`IOException` - if an I/O error occurs.

listSnapshots

```
public StateSnapshot[] listSnapshots()
  throws ClosedDocumentException
```

Returns the set of snapshots that are currently defined by invoking user for this document.

Returns:

the current set of snapshots.

Throws:

`ClosedDocumentException` - if this document is closed.

garbageCollect

```
public void garbageCollect(StateSnapshot snapshot)
  throws ClosedDocumentException,
         IncompatibleOpenModeException,
         java.io.FileNotFoundException,
         InvalidDocumentStateException,
         java.io.IOException
```

(continued from last page)

Garbage-collects the history of this document up to the specified snapshot. The specified snapshot must be in the current set of snapshots defined by invoking user. It must be also materialized and stable.

Parameters:

`snapshot` - the state up to which garbage-collect this document.

Throws:

`ClosedDocumentException` - if this document is closed.
`IncompatibleOpenModeException` - if this document is opened in read-only mode.
`FileNotFoundException` - if the specified state does not exist.
`InvalidDocumentStateException` - if the specified state is not stable.
`IOException` - if an I/O error occurs.

fr.inria.gforge.telex
Class IncompatibleOpenModeException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-fr.inria.gforge.telex.IncompatibleOpenModeException
```

All Implemented Interfaces:
java.io.Serializable

public class **IncompatibleOpenModeException**
extends java.lang.Exception

Thrown when a application writes to a document opened in read-only mode. The pathname of the document is provided in the detail message.
Author:
J-M. Busca INRIA/Regal

Field Summary

`public static final`	`serialVersionUID` Value: **1**

Constructor Summary

`public`	`IncompatibleOpenModeException(java.lang.String message)`

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

serialVersionUID

`public static final long` **`serialVersionUID`**

Constant value: **1**

Constructors

(continued from last page)

IncompatibleOpenModeException

```
public IncompatibleOpenModeException(java.lang.String message)
```

fr.inria.gforge.telex
Class InvalidDocumentStateException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-fr.inria.gforge.telex.InvalidDocumentStateException
```

All Implemented Interfaces:
java.io.Serializable

public class **InvalidDocumentStateException**
extends java.lang.Exception

Thrown when an application garbage-collects a document up to a state that is not valid. The name of the state is provided in the detail message.

Author:
J-M. Busca INRIA/Regal

Constructor Summary

public	InvalidDocumentStateException(java.lang.String message)

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

InvalidDocumentStateException

`public` **`InvalidDocumentStateException`**`(java.lang.String message)`

fr.inria.gforge.telex
Class InvalidFragmentException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-fr.inria.gforge.telex.InvalidFragmentException
```

All Implemented Interfaces:
java.io.Serializable

public class **InvalidFragmentException**
extends java.lang.Exception

Thrown when an application passes as parameter a fragment that is not valid. The id of the fragment is provided in the detail message.
Author:
J-M. Busca INRIA/Regal

Constructor Summary

public	InvalidFragmentException(java.lang.String message)

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

InvalidFragmentException

`public InvalidFragmentException(java.lang.String message)`

fr.inria.gforge.telex
Class InvalidScheduleException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-fr.inria.gforge.telex.InvalidScheduleException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **InvalidScheduleException**
extends java.lang.Exception

Thrown when an application passes as parameter a schedule that is not valid. The id of the schedule is provided in the detail message.
Author:
J-M. Busca INRIA/Regal

Constructor Summary

public	`InvalidScheduleException(java.lang.String message)`

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

InvalidScheduleException

`public InvalidScheduleException(java.lang.String message)`

fr.inria.gforge.telex Class InvalidSnapshotException

```
java.lang.Object
   |
   +-java.lang.Throwable
      |
      +-java.lang.Exception
         |
         +-fr.inria.gforge.telex.InvalidSnapshotException
```

All Implemented Interfaces:
java.io.Serializable

public class **InvalidSnapshotException**
extends java.lang.Exception

Thrown when an application passes as parameter a snapshot that is not valid. The id of the snapshot is provided in the detail message.
Author:
J-M. Busca INRIA/Regal

Constructor Summary

`public`	`InvalidSnapshotException(java.lang.String message)`

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

InvalidSnapshotException

`public InvalidSnapshotException(java.lang.String message)`

fr.inria.gforge.telex
Class Peer

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.Peer
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public class **Peer**
extends java.lang.Object
implements java.lang.Comparable, java.io.Serializable

The description of a `User` working at some `Site`. It consists of a reference to a user and a reference to a site. The *local* peer is the *invoking* user at the *local* site.

This class has a *natural ordering* consistent with `equals()`.
Author:
J-M. Busca INRIA/Regal

Method Summary	
`int`	`compareTo(Peer other)`
`static Peer`	`getInstance(java.lang.String user, java.lang.String site)` Returns the peer corresponding to the specified user id and site id.
`static Peer`	`getInstance(User user, Site site)` Returns the peer corresponding to the specified user and site.
`static Peer`	`getLocalPeer()` Returns the local peer.
`Site`	`getSite()` Returns the site part of this peer.
`java.lang.String`	`getSiteId()` Returns the site id part of this peer.
`User`	`getUser()` Returns the user part of this peer.
`java.lang.String`	`getUserId()` Returns the user id part of this peer.
`boolean`	`isLocal()` Returns whether this peer is the local peer.
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`
`compareTo`

Methods

getInstance

```
public static Peer getInstance(User user,
         Site site)
```

Returns the peer corresponding to the specified user and site.

Parameters:
`user` - the user part of the peer.
`site` - the site part of the peer.

Returns:
the peer corresponding to the specified user and site.

getInstance

```
public static Peer getInstance(java.lang.String user,
         java.lang.String site)
```

Returns the peer corresponding to the specified user id and site id.

Parameters:
`user` - the user part of the peer.
`site` - the site part of the peer.

Returns:
the peer corresponding to the specified user id and site id.

toString

```
public java.lang.String toString()
```

getUser

```
public User getUser()
```

Returns the user part of this peer.

Returns:
the user part of this peer.

getUserId

```
public java.lang.String getUserId()
```

Returns the user id part of this peer.

Returns:

(continued from last page)

the user id part of this peer.

getSite

`public Site getSite()`

Returns the site part of this peer.

Returns:
the site part of this peer.

getSiteId

`public java.lang.String getSiteId()`

Returns the site id part of this peer.

Returns:
the site id part of this peer.

isLocal

`public boolean isLocal()`

Returns whether this peer is the local peer.

Returns:
true if this peer is the local peer and false otherwise.

getLocalPeer

`public static Peer getLocalPeer()`

Returns the local peer.

Returns:
the local peer.

compareTo

`public int compareTo(Peer other)`

fr.inria.gforge.telex
Interface Schedule

public interface **Schedule**
extends

A sequence of `Action`s to be applied on a Telex `Document`. It is defined by the sequence of actions to apply and the `DocumentState` to start from. A schedule is computed by a `ScheduleGenerator` from the action-constraint graph representing the document. If some actions of the graph are antagonistic, then a schedule may not contain all of them. Actions that are excluded from the schedule, or *non-actions* are provided for information purposes.

A schedule is identified by an id of type String assigned by Telex. When processing bound documents, Telex assigns the same id to related per-document schedule. Schedules are immutable objects.

Author:
J-M. Busca INRIA/Regal

Method Summary

`Action[]`	`getActions()` Returns the sequence of actions corresponding to this schedule.
`Schedule[]`	`getBoundSchedules()` Returns the list of schedules bound with this schedule.
`Document`	`getDocument()` Returns the document this schedule relates to.
`java.lang.String`	`getId()` Returns the id of this schedule.
`Action[]`	`getNonActions()` Returns the set of action that are excluded from this schedule.
`DocumentState`	`getState()` Returns the document state this schedule must be applied on.

Methods

getState

`public DocumentState getState()`

Returns the document state this schedule must be applied on. Telex keeps a copy of this state and returns a fresh new copy every time this method is called. The application may thus safely update the returned state object when applying the actions of the schedule.

As a special case, if (i) the state is the initial state specified in the `ProcessingParameters`s and (ii) the `DocumentState` class implements the rebuild() method, then this method calls the rebuild() method on the initial state and returns it. No copy actually takes places.

Returns:
the document state this schedule must be applied on.

(continued from last page)

getActions

`public Action[] getActions()`

Returns the sequence of actions corresponding to this schedule. Note that in special cases, e.g. bound document processing, the sequence may be empty. The implied state of the document is then the state returned by `getState()`.

Returns:
the sequence of actions corresponding to this schedule.

getNonActions

`public Action[] getNonActions()`

Returns the set of action that are excluded from this schedule. These action are excluded because they conflict with actions that belong to this schedule.

Returns:
the set of action that are excluded from this schedule.

getId

`public java.lang.String getId()`

Returns the id of this schedule. Related per-document schedules share the same id.

Returns:
the id of this schedule.

getDocument

`public Document getDocument()`

Returns the document this schedule relates to.

Returns:
the document this schedule relates to.

getBoundSchedules

`public Schedule[] getBoundSchedules()`

Returns the list of schedules bound with this schedule. The list excludes this schedule. If this schedule is not bound with any other schedule, the method returns an array of size 0.

Returns:
the list of schedules that are bound with this one.

fr.inria.gforge.telex
Class Site

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.Site
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public class **Site**
extends java.lang.Object
implements java.lang.Comparable, java.io.Serializable

The description of a site hosting one or more `User`s. Each site is uniquely identified throughout the system by an id of type `java.lang.String`. The *local* site is the site this Virtual Machine is running on.

This class has a *natural ordering* consistent with `equals()`.
See Also:
`User`, `Peer`
Author:
Abhishek Gupta INRIA/Regal, J-M. Busca INRIA/Regal

Method Summary

`int`	`compareTo(Site to)`
`long`	`getDistance()` Returns the distance from local site to this site.
`java.lang.String`	`getId()` Returns the unique id of this site.
`static Site`	`getInstance(java.lang.String id)` Returns the description of the site with the specified id.
`static Site`	`getLocalSite()` Returns the description of the local site.
`boolean`	`isLocal()` Returns whether this site is the local site.
`void`	`setDistance(long distance)` **Deprecated.**
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Methods

getInstance

`public static Site getInstance(java.lang.String id)`

Returns the description of the site with the specified id.

Parameters:
`id` - the id of the site to look up.

Returns:
the description of the site with the specified id.

toString

`public java.lang.String toString()`

getId

`public java.lang.String getId()`

Returns the unique id of this site.

Returns:
the id of this site.

isLocal

`public boolean isLocal()`

Returns whether this site is the local site.

Returns:
true if this site is the local site and false otherwise.

getDistance

`public long getDistance()`

Returns the distance from local site to this site. The distance is defined as the network delay between the two sites, measured in microseconds.

Returns:
the distance from local site to this site.

setDistance

`public void setDistance(long distance)`

Deprecated.

(continued from last page)

Sets the distance from local site to this site to the specified value. The distance is defined as the network delay between the two sites.

This method is for Telex's internal use only.

Parameters:

`distance` - the new distance value.

getLocalSite

```
public static Site getLocalSite()
```

Returns the description of the local site.

Returns:

the description of the local site.

compareTo

```
public int compareTo(Site to)
```

fr.inria.gforge.telex
Class Telex

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.Telex
```

public abstract class **Telex**
extends java.lang.Object

The main class of the Telex middleware. Several `TelexApplication` may run concurrently within a single Virtual Machine and use the services of Telex. Each application must first create a instance of Telex associated with it. Using this instance, the application may then open Telex `Document`s.

An application may process documents of various *types*. The type of a document is defined by the suffix of its name, e.g. "tdoc" without the '.' character. Telex allows an application to associate some `ProcessingParameters` with each of the document types it handles. When opening a document, Telex will apply to the document the parameters corresponding to its type. Note that the same document type may be registered by several applications in their respective Telex instance.

Each application may open one or more documents. The number of documents that can be opened is only limited by system ressources, mainly memory. A document that already exists may be open in either `Telex.OpenMode.READ_ONLY` or `Telex.OpenMode.READ_WRITE` mode. A new document may be created by opening it in `Telex.OpenMode.CREATE` mode. All opened documents are automatically closed when the Virtual Machine exits.

All applications running in a Virtual Machine execute on behalf of the user who invoked the VM. All the actions and all the constraints that these applications create will be attributed to this user. All Telex documents that these applications create will be owned by this user.

The Telex middleware may operate in two modes, as defined by the telex.mode java property:

- *Stand-alone* mode (telex.mode="standalone"). In this mode, Telex peers communicate through a embedded communication library. Each peer is responsible for keeping its own replica of the multi-log up-to-date.
- *Grid4All* mode (telex.mode="grid4all"). In this mode, Telex relies on the underlying distributed file system for storing and replicating multi-logs, and for transmitting messages between Telex peers.

Author:
J-M. Busca INRIA/Regal

Nested Class Summary

`class`	`Telex.OpenMode` Telex.OpenMode

Constructor Summary

`protected`	`Telex()`

Method Summary

`static Telex`	`getInstance(TelexApplication application)` Creates a Telex instance for the specified Telex application.
`static Telex`	`getInstance(TelexApplication application, ProcessingParameters parameters)` Creates a Telex instance for the specified Telex application and registers its default processing parameters.

`Document`	`openDocument(java.lang.String pathname)` Opens the Telex document with the specified pathname in `READ_WRITE` mode.
`abstract Document`	`openDocument(java.lang.String pathname, Document target)` Opens the Telex document with the specified pathname and merges its actions and constraints with the specified document.
`abstract Document`	`openDocument(java.lang.String pathname, Telex.OpenMode mode)` Opens the Telex document with the specified pathname in the specified mode.
`abstract boolean`	`registerType(java.lang.String type, ProcessingParameters parameters)` Registers the specified document type in this Telex instance and associates it with the specified processing parameters.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Telex

```
protected Telex()
```

Methods

getInstance

```
public static Telex getInstance(TelexApplication application)
```

Creates a Telex instance for the specified Telex application. No document type is registered in this Telex instance after this method returns. The application must define at least one document type by calling the `registerType(String, ProcessingParameters)` method to be allowed to open documents.

Parameters:

`application` - the Telex application that registers.

Returns:

a Telex instance associated with application.

getInstance

```
public static Telex getInstance(TelexApplication application,
         ProcessingParameters parameters)
```

(continued from last page)

Creates a Telex instance for the specified Telex application and registers its default processing parameters. This is a convenience method that is equivalent to:

```
Telex instance = Telex.getInstance(application);
instance.registerType(DEFAULT_TYPE, parameters);
```

Parameters:

`application` - the Telex application that registers.

`parameters` - the default processing parameters for application.

Returns:

a Telex instance associated with application.

registerType

```
public abstract boolean registerType(java.lang.String type,
         ProcessingParameters parameters)
```

Registers the specified document type in this Telex instance and associates it with the specified processing parameters. If the specified type is `Document.DEFAULT_TYPE`, the specified parameters will be applied to documents whose type is not otherwise registered. A given document type can only be registered once: subsequent attempts to register it again will fail, as indicated by the value returned by this method.

Parameters:

`type` - the document type to register, e.g. "tdoc".

`parameters` - the processing parameters for type.

Returns:

true if the type was registered successfully, and false if the type was already registered.

openDocument

```
public Document openDocument(java.lang.String pathname)
  throws UnknownDocumentTypeException,
         java.io.FileNotFoundException,
         java.io.IOException
```

Opens the Telex document with the specified pathname in `READ_WRITE` mode.

Parameters:

`pathname` - the pathname of the document.

Returns:

the corresponding `Document` object.

Throws:

`UnknownDocumentTypeException` - if the type of the document is not registered.

`FileNotFoundException` - if the document does not exist and open mode is not CREATE.

`IOException` - if another I/O error occurs.

(continued from last page)

openDocument

```
public abstract Document openDocument(java.lang.String pathname,
         Telex.OpenMode mode)
  throws UnknownDocumentTypeException,
         java.io.FileNotFoundException,
         java.io.IOException
```

Opens the Telex document with the specified pathname in the specified mode. The type of the document, or the `Document.DEFAULT_TYPE` type, must be registered for the call to succeed. The specified document will be processed according to the parameters associated with its type.

Parameters:

`pathname` - the pathname of the document.

`mode` - the mode in which to open the document.

Returns:

the corresponding `Document` object.

Throws:

`UnknownDocumentTypeException` - if the type of the document is not registered.

`FileNotFoundException` - if the document is not found.

`IOException` - if an I/O error occurs.

openDocument

```
public abstract Document openDocument(java.lang.String pathname,
         Document target)
  throws UnknownDocumentTypeException,
         java.io.FileNotFoundException,
         java.io.IOException
```

Opens the Telex document with the specified pathname and merges its actions and constraints with the specified document. The new document is opened with the mode defined for the specified document. The type of the document, or the `Document.DEFAULT_TYPE` type, must be registered for the call to succeed. The specified document will be processed according to the parameters associated with its type.

Parameters:

`pathname` - the pathname of the document.

`target` - the document to merge the new document with.

Returns:

the corresponding `Document` object.

Throws:

`UnknownDocumentTypeException` - if the type of the document is not registered.

`FileNotFoundException` - if the document is not found.

`IOException` - if an I/O error occurs.

fr.inria.gforge.telex
Class Telex.OpenMode

```
java.lang.Object
    |
    +-java.lang.Enum
        |
        +-fr.inria.gforge.telex.Telex.OpenMode
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **Telex.OpenMode**
extends java.lang.Enum

The open mode of a Telex `Document`.

Field Summary

`public static final`	`CREATE` Create the specified document if it does not exist, and open it in read-write mode.
`public static final`	`READ_ONLY` Open the specified document in read-only mode.
`public static final`	`READ_WRITE` Open the specified document in read-write mode.

Method Summary

`static Telex.OpenMode`	`valueOf(java.lang.String name)`
`static Telex.OpenMode[]`	`values()`

Methods inherited from class `java.lang.Enum`

`clone, compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Fields

(continued from last page)

READ_ONLY

`public static final fr.inria.gforge.telex.Telex.OpenMode` **`READ_ONLY`**

Open the specified document in read-only mode.

READ_WRITE

`public static final fr.inria.gforge.telex.Telex.OpenMode` **`READ_WRITE`**

Open the specified document in read-write mode.

CREATE

`public static final fr.inria.gforge.telex.Telex.OpenMode` **`CREATE`**

Create the specified document if it does not exist, and open it in read-write mode.

Methods

values

`public final static Telex.OpenMode[]` **`values`**`()`

valueOf

`public static Telex.OpenMode` **`valueOf`**`(java.lang.String name)`

fr.inria.gforge.telex
Class UninstantiableClassException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-fr.inria.gforge.telex.UninstantiableClassException
```

All Implemented Interfaces:
java.io.Serializable

public class **UninstantiableClassException**
extends java.lang.Exception

Thrown when an application provides a parameter class that can not be instantiated. The name of the class is provided in the detail message.

Author:
J-M. Busca INRIA/Regal

Constructor Summary

public	UninstantiableClassException(java.lang.String message)

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

UninstantiableClassException

```
public UninstantiableClassException(java.lang.String message)
```

fr.inria.gforge.telex
Class UnknownDocumentTypeException

```
java.lang.Object
  |
  +-java.lang.Throwable
      |
      +-java.lang.Exception
          |
          +-fr.inria.gforge.telex.UnknownDocumentTypeException
```

All Implemented Interfaces:
java.io.Serializable

public class **UnknownDocumentTypeException**
extends java.lang.Exception

Thrown when an application accesses a document whose type is not registered. The pathname of the document is provided in the detail message.
Author:
J-M. Busca INRIA/Regal

Constructor Summary

`public`	`UnknownDocumentTypeException(java.lang.String message)`

Methods inherited from class `java.lang.Throwable`

`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

UnknownDocumentTypeException

`public UnknownDocumentTypeException(java.lang.String message)`

fr.inria.gforge.telex
Class User

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.User
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable, java.security.Principal

---

public class **User**
extends java.lang.Object
implements java.security.Principal, java.lang.Comparable, java.io.Serializable

The description of a user editing a `Document`. Each user is uniquely identified throughout the system by an id of type `java.lang.String`. The *invoking* user is the user who invoked this Virtual Machine. This user will be attributed all actions created within this VM and he will own all documents created within this VM.

This class has a *natural ordering* consistent with `equals()`.
See Also:
`Site`, `Peer`
Author:
J-M. Busca INRIA/Regal

Method Summary

`int`	`compareTo(User to)`
`java.lang.String`	`getId()` Returns the id of this user.
`static User`	`getInstance(java.lang.String id)` Returns the description of the user with the specified id.
`static User`	`getInvokingUser()` Returns the description of the invoking user.
`java.lang.String`	`getName()` Returns the name of this user.
`boolean`	`isInvoking()` Returns whether this user is the invoking user.
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.security.Principal`

`equals, getName, hashCode, toString`

Methods inherited from interface `java.lang.Comparable` `compareTo`

Methods

getInstance

`public static User getInstance(java.lang.String id)`

Returns the description of the user with the specified id.

Parameters:
`id` - the id of the user to look up.

Returns:
the description of the user with the specified id.

toString

`public java.lang.String toString()`

getId

`public java.lang.String getId()`

Returns the id of this user.

Returns:
the id of this user.

getName

`public java.lang.String getName()`

Returns the name of this user. This method makes up the `java.security.Principal` interface.

Returns:
the name of this user.

isInvoking

`public boolean isInvoking()`

Returns whether this user is the invoking user.

Returns:
true if this user is the invoking user and false otherwise.

getInvokingUser

`public static User getInvokingUser()`

Returns the description of the invoking user.

(continued from last page)

Returns:
the description of inovking user.

compareTo

```
public int compareTo(User to)
```

Package fr.inria.gforge.telex.application

Provides classes and interfaces of objects instantiated by Telex applications.

fr.inria.gforge.telex.application
Class Action

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.Action
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public class **Action**
extends java.lang.Object
implements java.lang.Comparable, java.io.Serializable

The generic representation of an action in Telex. As defined by the ACF, an action represent an operation on a `Document`. It has several generic attributes that applications may use as `ActionFilter` criteria. It also defines the application-specific keys attribute, which Telex uses to determine whether to call the `ConstraintChecker` associated to the document. All of these attributes are immutable and except for the key attribute, they are all set after a successful call to `Document.addFragment(Fragment)`. An action is uniquely identified by the tuple of attributes (document, issuer, location, time-stamp).

Unlike `Constraint`s, an action may belong to only one document. See the `Constraint` class on how constraints between actions are generated.

The *keys* of an action are a set of int values representing the application object that the action targets. Action keys are used to improve performance of constraint checking by avoiding unnecessary calls to the constraint checker. Given two actions of the same document, issued by distinct users, Telex calls the `ConstraintChecker.getConstraints(Action, Action)` method if the key sets of the two actions intersect.

This class is meant to be sub-classed by Telex applications in order to define application-specific attributes, such as the operation that the action represents, its parameters, etc. These attributes will be stored in the multi-log along the generic attributes and they will be transmitted to other peer Telex instances.
Author:
J-M. Busca INRIA/Regal, Abhishek Gupta INRIA/Regal

Nested Class Summary	
`class`	`Action.CommitStatus` Action.CommitStatus
`class`	`Action.SerializationStatus` Action.SerializationStatus

Constructor Summary	
`public`	`Action()` Creates a new action with no associated action key.
`public`	`Action(int[] keys)` Creates a new action associated with the specified keys.
`public`	`Action(ActionId id)` **Deprecated.**

Method Summary

void	addUnresolvedConstraint(Constraint unresolved) **Deprecated.**
int	compareTo(Action other) Compares this action with the specified action for happened-before order.
boolean	equals(java.lang.Object object)
Action.CommitStatus	getCommitStatus() Returns the commit status of this action.
Document	getDocument() Returns the Telex document this action must be / is written to.
ActionId	getId() Returns the id of this action.
User	getIssuer() Returns the user who issued this action.
int[]	getKeys() Returns the keys associated with this action.
Site	getLocation() Returns the site where this action was issued.
Peer	getPeer() Returns the peer who issued this action.
Action.SerializationStatus	getSerializationStatus() Returns the serialization status of this action.
long	getTime() Returns the time this action was submitted.
long	getTimestamp() Returns the time-stamp of this action.
java.util.Set	getUnresolvedConstraints() Returns the set of unresolved constraints of this action.
ValueVector.VersionVector	getVersionVector() **Deprecated.**
long	getWeight()
boolean	isAborted() Returns whether this action is aborted.
boolean	isCommitted() Returns whether this action is committed.
boolean	isDecided() Returns whether this action is decided.
boolean	isLocal() Returns whether this action was issued locally.

`boolean`	`isStable()` Returns whether this action is stable.
`void`	`read(java.io.DataInput in)` Reads this action from the specified input.This method is meant to be overridden by subclasses in order to write their own fields.
`static Action[]`	`readActions(DocumentImpl document, java.io.DataInput in)` Reads a sequence of actions of the specified document from the specified input.
`java.lang.Object`	`readResolve()`
`void`	`removeUnresolvedConstraint(Constraint resolved)` **Deprecated.**
`void`	`setCommitStatus(Action.CommitStatus status)` Sets the commit status of this action to the specified value.
`void`	`setDocument(Document document)` Sets the Telex document this action must be written to.
`void`	`setId(ActionId id)` **Deprecated.**
`void`	`setSerializationStatus(Action.SerializationStatus status)` **Deprecated.**
`void`	`setTime(long time)` **Deprecated.**
`void`	`setTimestamp(long timestamp)` **Deprecated.**
`void`	`setVersionVector(ValueVector.VersionVector vector)` **Deprecated.**
`void`	`setWeight(long w)`
`java.lang.String`	`toString()`
`void`	`write(java.io.DataOutput out)` Writes this action to the specified output.
`static void`	`writeActions(Action[] actions, java.io.DataOutput out)` Writes the specified sequence of actions to the specified output.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Constructors

(continued from last page)

Action

`public **Action**()`

Creates a new action with no associated action key.

Action

`public **Action**(int[] keys)`

Creates a new action associated with the specified keys.

Parameters:
`keys` - the set of keys to associate the action with.

Action

`public **Action**(ActionId id)`

Deprecated.

Creates a new action with the specified id.

For Telex's internal use only.

Parameters:
`id` - the id of the new action.

Methods

toString

`public java.lang.String **toString**()`

equals

`public final boolean **equals**(java.lang.Object object)`

compareTo

`public int **compareTo**(Action other)`

Compares this action with the specified action for happened-before order. Actions submitted within the same fragment compare as 0, i.e. are concurrent. Note that The method throws a RuntimeException if the computation of the happened-before relation is disabled.

Parameters:
`other` - the action to compare this action with.

Returns:
a negative integer, zero or a positive integer as this action happened before, after or is concurrent to the specified action.

(continued from last page)

getKeys

`public final int[] getKeys()`

Returns the keys associated with this action. If this action is not associated with any action key, this method returns null. This attribute is set by Action constructors.

Returns:
the keys associated with this action, or null if no action key is associated with this action.

getTime

`public final long getTime()`

Returns the time this action was submitted. It is the value read on the system clock of the site that issued this action. This attribute is meant to be used by `ActionFilter`s. It should be considered with caution since the system clocks of cooperating sites may not be accurate and/or synchronized.

This attribute is set after a successful call to `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is -1.

Returns:
the time this action was generated.

setTime

`public final void setTime(long time)`

Deprecated.

Sets the time this action was submitted.

For Telex's internal use only.

Parameters:
`time` - the time to set.

getTimestamp

`public final long getTimestamp()`

Returns the time-stamp of this action. It is the sequence number of this action relative to the issuer peer, starting from 0. This attributes is meant to be used by `ActionFilter`s.

This attribute is set after a successful call to `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is less than 0.

Returns:
the time-stamp of this action, or -1 if no call to addFragment() with this action as parameter ever succeeded.

setTimestamp

`public final void setTimestamp(long timestamp)`

Deprecated.

Sets the time-stamp of this action.

For Telex's internal use only.

Parameters:

(continued from last page)

`timestamp` - the time-stamp to set.

---

getVersionVector

`public final ValueVector.VersionVector **getVersionVector**()`

Deprecated.

Returns the version vector of this action.

For Telex's internal use only.

Returns:
the version vector of this action.

---

setVersionVector

`public final void **setVersionVector**(ValueVector.VersionVector vector)`

Deprecated.

Sets the version vector of this action.

For Telex's internal use only.

Parameters:
`vector` - the version vector to set.

---

getId

`public final ActionId **getId**()`

Returns the id of this action. The id uniquely identifies this action across all sites.

This attribute is set after a successful call to `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is null.

Returns:
the id of this action.

---

setId

`public final void **setId**(ActionId id)`

Deprecated.

Sets the id of this action.

For Telex's internal use only.

Parameters:
`id` - the id to give this action.

---

getPeer

`public final Peer **getPeer**()`

Returns the peer who issued this action.

This attribute is set after a successful call to `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is null.

(continued from last page)

Returns:
the peer who issued this action.

getIssuer

`public final User getIssuer()`

Returns the user who issued this action.

This attribute is set after a successful call to `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is null.

Returns:
the user who issued this action.

getLocation

`public final Site getLocation()`

Returns the site where this action was issued.

This attribute is set after a successful call to `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is null.

Returns:
the site where this action was issued.

isLocal

`public final boolean isLocal()`

Returns whether this action was issued locally.

This attribute is set after a successful call to `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is undefined.

Returns:
true if this action was issued locally and false otherwise.

setDocument

`public void setDocument(Document document)`

Sets the Telex document this action must be written to. This method must be called prior to submitting this action through the `addOverlapingFragment(Fragment)` method.

This method cannot be called after this actions has been written to a document.

Parameters:
`document` - the document this action must be written to.

getDocument

`public Document getDocument()`

Returns the Telex document this action must be / is written to.

This attribute is set after a successful call to `setDocument(Document)` or `Document.addFragment(Fragment)` with this action as parameter. Before such a call, this attribute is null.

(continued from last page)

Returns:
the Telex document this action must be / is written to.

getWeight

`public long getWeight()`

setWeight

`public void setWeight(long w)`

addUnresolvedConstraint

`public void addUnresolvedConstraint(Constraint unresolved)`

Deprecated.

Adds the specified constraint to the set of unresolved constraints of this action. A constraint is unresolved if it targets an action of a document that is not merged to the document this action belongs to.

This methods is for Telex's internal use only.

Parameters:
`unresolved` - the unresolved constraint to add.

removeUnresolvedConstraint

`public void removeUnresolvedConstraint(Constraint resolved)`

Deprecated.

Removes the specified constraint from the set of unresolved constraints of this action. A constraint is unresolved if it targets an action of a document that is not merged to the document this action belongs to.

This methods is for Telex's internal use only.

Parameters:
`resolved` - the new set of unresolved constraints.

getUnresolvedConstraints

`public java.util.Set getUnresolvedConstraints()`

Returns the set of unresolved constraints of this action. A constraint is unresolved if it targets an action of a document that is not merged to the document this action belongs to.

Returns:
the set of unresolved constraints involving this action.

isCommitted

`public boolean isCommitted()`

Returns whether this action is committed.

Returns:

(continued from last page)

whether this action is committed.

isAborted

`public boolean isAborted()`

Returns whether this action is aborted.

Returns:
whether this action is aborted.

isDecided

`public boolean isDecided()`

Returns whether this action is decided.

Returns:
whether this action is decided.

isStable

`public boolean isStable()`

Returns whether this action is stable.

Returns:
whether this action is stable.

getCommitStatus

`public Action.CommitStatus getCommitStatus()`

Returns the commit status of this action.

Returns:
the commit status of this action.

getSerializationStatus

`public Action.SerializationStatus getSerializationStatus()`

Returns the serialization status of this action.

Returns:
the serialization status of this action.

setCommitStatus

`public void setCommitStatus(Action.CommitStatus status)`

Sets the commit status of this action to the specified value.

This methods is for Telex's internal use only.

Parameters:
`status` - the new commit status of this action.

(continued from last page)

setSerializationStatus

```
public void setSerializationStatus(Action.SerializationStatus status)
```

Deprecated.

Sets the serialization status of this action to the specified value.

This methods is for Telex's internal use only.

Parameters:
`status` - the new serialization status of this action.

---

write

```
public void write(java.io.DataOutput out)
  throws java.io.IOException
```

Writes this action to the specified output. This method is meant to be overridden by subclasses in order to write their own fields. The overridden method **must call** this method with super.write() as the **first** statement.

Parameters:
`out` - the output to write this action to.

Throws:
`IOException` - if an I/O error occurs.

---

read

```
public void read(java.io.DataInput in)
  throws java.io.IOException
```

Reads this action from the specified input.This method is meant to be overridden by subclasses in order to write their own fields. The overridden method **must call** this method with super.read() as the **first** statement.

Parameters:
`in` - the input to read this action from.

Throws:
`IOException` - if an I/O error occurs.

---

writeActions

```
public static void writeActions(Action[] actions,
         java.io.DataOutput out)
  throws java.io.IOException
```

Writes the specified sequence of actions to the specified output. The method compresses action representation by using a translation table, which is also written to the buffer.

Parameters:
`actions` - the sequence of actions to write.
`out` - the output to write the sequence of actions to.

Throws:
`IOException` - if an I/O error occurs.

(continued from last page)

readActions

```
public static Action[] readActions(DocumentImpl document,
         java.io.DataInput in)
  throws java.io.IOException
```

Reads a sequence of actions of the specified document from the specified input. The sequence must have been written by the `writeActions(Action[], DataOutput)` method.

Parameters:

`document` - the document the actions belong to.

`in` - the input to read the sequence of actions from.

Returns:

the read sequence of actions.

Throws:

`IOException` - if an I/O error occurs.

readResolve

```
public java.lang.Object readResolve()
```

fr.inria.gforge.telex.application
Class Action.CommitStatus

```
java.lang.Object
  |
  +-java.lang.Enum
      |
      +-fr.inria.gforge.telex.application.Action.CommitStatus
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

public static final class **Action.CommitStatus**
extends java.lang.Enum

The status of an action with respect to commitment.

Field Summary

`public static final`	`ABORTED` The action is a non-action of every schedule.
`public static final`	`GUARANTEED` The action is an action of every schedule.
`public static final`	`UNDEFINED` The status of the action is not defined yet.

Method Summary

`static Action.CommitStatus`	`valueOf(java.lang.String name)`
`static Action.CommitStatus[]`	`values()`

Methods inherited from class `java.lang.Enum`

`clone, compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Fields

(continued from last page)

UNDEFINED

`public static final fr.inria.gforge.telex.application.Action.CommitStatus` **`UNDEFINED`**

The status of the action is not defined yet.

ABORTED

`public static final fr.inria.gforge.telex.application.Action.CommitStatus` **`ABORTED`**

The action is a non-action of every schedule.

GUARANTEED

`public static final fr.inria.gforge.telex.application.Action.CommitStatus` **`GUARANTEED`**

The action is an action of every schedule.

Methods

values

`public final static Action.CommitStatus[]` **`values`**`()`

valueOf

`public static Action.CommitStatus` **`valueOf`**`(java.lang.String name)`

fr.inria.gforge.telex.application
Class Action.SerializationStatus

```
java.lang.Object
  |
  +-java.lang.Enum
      |
      +-fr.inria.gforge.telex.application.Action.SerializationStatus
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **Action.SerializationStatus**
extends java.lang.Enum

The status of an action with respect to serialization.

Field Summary

public static final	SERIALIZED The action is GUARANTEED and ordered with respect to all non-commuting GUARANTEED actions.
public static final	STABLE The action is either ABORTED, or GUARANTEED and SERIALIZED and all preceding actions are STABLE.
public static final	UNKNOWN The status of the action is not defined yet.

Method Summary

static Action.SerializationStatus	valueOf(java.lang.String name)
static Action.SerializationStatus[]	values()

Methods inherited from class `java.lang.Enum`

`clone, compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

(continued from last page)

Fields

UNKNOWN

`public static final fr.inria.gforge.telex.application.Action.SerializationStatus` **`UNKNOWN`**

The status of the action is not defined yet.

SERIALIZED

`public static final fr.inria.gforge.telex.application.Action.SerializationStatus` **`SERIALIZED`**

The action is GUARANTEED and ordered with respect to all non-commuting GUARANTEED actions.

STABLE

`public static final fr.inria.gforge.telex.application.Action.SerializationStatus` **`STABLE`**

The action is either ABORTED, or GUARANTEED and SERIALIZED and all preceding actions are STABLE.

Methods

values

`public final static Action.SerializationStatus[]` **`values`**`()`

---

valueOf

`public static Action.SerializationStatus` **`valueOf`**`(java.lang.String name)`

fr.inria.gforge.telex.application
Class ActionFilter

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ActionFilter
```

All Implemented Interfaces:
java.io.Serializable

Direct Known Subclasses:
RetainThisDocumentActionsOnly, ExcludeTheseUsersActions, RetainTheseUsersActionsOnly, RetainMyActionsOnly

public abstract class **ActionFilter**
extends java.lang.Object
implements java.io.Serializable

An object that specifies `Action`s to exclude from the view of a `Document`. An action filter may be associated with one or more documents by calling the `Document.defineFilter(ActionFilter)` method. An action filter has a name for convenience; Telex does not use it.

An action filter may be activated or de-activated by calling the `setActive(boolean)` method. The change of activity status applies to all documents that the filter is associated with. When generating sound `Schedule`s on the document, only the active filters of the document are applied. The activity status of an action filter is saved as part of the persistent state of each of the documents it is associated with when closing the document.

This class may be sub-classed by applications, for instance to provide methods for describing a filter. A filter must be serializable in order to be saved as part of the persistent state of a document. This class provides several filters of general interest, as the `ActionFilter.RetainMyActionsOnly` filter.
Author:
P. Sutra INRIA/Regal, J-M. Busca INRIA/Regal

Nested Class Summary

`class`	`ActionFilter.ExcludeTheseUsersActions` ActionFilter.ExcludeTheseUsersActions
`class`	`ActionFilter.RetainMyActionsOnly` ActionFilter.RetainMyActionsOnly
`class`	`ActionFilter.RetainTheseUsersActionsOnly` ActionFilter.RetainTheseUsersActionsOnly
`class`	`ActionFilter.RetainThisDocumentActionsOnly` ActionFilter.RetainThisDocumentActionsOnly

Field Summary

`protected transient`	`_document`
`protected`	`_isActive`
`protected`	`_name`

Constructor Summary

public	`ActionFilter(java.lang.String name)` Creates an active action filter with the specified name.
public	`ActionFilter(java.lang.String name, boolean active)` Creates an action filter with the specified name and activity status.

Method Summary

`java.lang.String`	`getName()` Returns the name of this action filter.
`boolean`	`isActive()` Determines whether this action filter is active.
`abstract boolean`	`isFiltered(Action a)` Determines whether the specified action must be filtered out.
`void`	`setActive(boolean on)` Sets the activity status of this action filter to the specified value.
`void`	`setDocument(Document document)` Associate this filter with the specified document.
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

_name

`protected java.lang.String` **`_name`**

_isActive

`protected boolean` **`_isActive`**

_document

`protected transient fr.inria.gforge.telex.Document` **`_document`**

Constructors

(continued from last page)

ActionFilter

```
public ActionFilter(java.lang.String name)
```

Creates an active action filter with the specified name.

Parameters:
`name` - the name of the new action filter.

ActionFilter

```
public ActionFilter(java.lang.String name,
                    boolean active)
```

Creates an action filter with the specified name and activity status.

Parameters:
`name` - the name of the new action filter.
`active` - the activity status of the new action filter.

Methods

toString

```
public java.lang.String toString()
```

---

getName

```
public java.lang.String getName()
```

Returns the name of this action filter.

Returns:
the name of this action filter.

---

isActive

```
public final boolean isActive()
```

Determines whether this action filter is active.

Returns:
true is this action filter is active, and false otherwise.

---

setActive

```
public final void setActive(boolean on)
```

Sets the activity status of this action filter to the specified value.

Parameters:
`on` - true is this filter must be activated, and false if this filter must be de-activated.

(continued from last page)

setDocument

`public final void setDocument(Document document)`

Associate this filter with the specified document. Telex calls this method when the `Document.defineFilter(ActionFilter)` is called on the specified document with this filter as parameter. It also calls this methods when this filter is restored from persistent storage.

Parameters:
`document` - the document to associate this filter with.

isFiltered

`public abstract boolean isFiltered(Action a)`

Determines whether the specified action must be filtered out. This method is meant to be overridden by sub-classes: no default implementation is provided.

Parameters:
`a` - the action to check.

Returns:
true if the action must be filtered out, and false otherwise.

fr.inria.gforge.telex.application
Class ActionFilter.RetainMyActionsOnly

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ActionFilter
      |
      +-fr.inria.gforge.telex.application.ActionFilter.RetainMyActionsOnly
```

All Implemented Interfaces:
java.io.Serializable

---

public static class **ActionFilter.RetainMyActionsOnly**
extends ActionFilter

An action filter that retains only the actions that are issued by invoking user.
Author:
J-M. Busca INRIA/Regal

Fields inherited from class `fr.inria.gforge.telex.application.ActionFilter`

`_document`, `_isActive`, `_name`

Constructor Summary

`public`	`ActionFilter.RetainMyActionsOnly(java.lang.String name)` An action filter that retains only the actions that are issued by invoking user.

Method Summary

`boolean`	`isFiltered(Action a)`
`java.lang.String`	`toString()`

Methods inherited from class `fr.inria.gforge.telex.application.ActionFilter`

`getName`, `isActive`, `isFiltered`, `setActive`, `setDocument`, `toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

ActionFilter.RetainMyActionsOnly

`public ActionFilter.RetainMyActionsOnly(java.lang.String name)`

An action filter that retains only the actions that are issued by invoking user.

Parameters:

(continued from last page)

`name` - the name of the filter.

Methods

toString

```
public java.lang.String toString()
```

isFiltered

```
public final boolean isFiltered(Action a)
```

Determines whether the specified action must be filtered out. This method is meant to be overridden by sub-classes: no default implementation is provided.

fr.inria.gforge.telex.application
Class ActionFilter.RetainTheseUsersActionsOnly

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ActionFilter
      |
      +-fr.inria.gforge.telex.application.ActionFilter.RetainTheseUsersActionsOnly
```

All Implemented Interfaces:
java.io.Serializable

public static class **ActionFilter.RetainTheseUsersActionsOnly**
extends ActionFilter

An action filter that retains only the actions that are issued by a set of users.
Author:
J-M. Busca INRIA/Regal

Fields inherited from class fr.inria.gforge.telex.application.ActionFilter
_document, _isActive, _name

Constructor Summary	
public	ActionFilter.RetainTheseUsersActionsOnly(java.lang.String name, java.util.Collection users) Creates an action filter that retains only the actions that are issued by the specified set of users.

Method Summary	
boolean	isFiltered(Action a)
java.lang.String	toString()

Methods inherited from class fr.inria.gforge.telex.application.ActionFilter
getName, isActive, isFiltered, setActive, setDocument, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

ActionFilter.RetainTheseUsersActionsOnly

```
public ActionFilter.RetainTheseUsersActionsOnly(java.lang.String name,
                                                java.util.Collection users)
```

Creates an action filter that retains only the actions that are issued by the specified set of users.

(continued from last page)

Parameters:

`name` - the name of the filter.

`users` - the list of users whose actions must be retained.

Methods

toString

```
public java.lang.String toString()
```

---

isFiltered

```
public final boolean isFiltered(Action a)
```

Determines whether the specified action must be filtered out. This method is meant to be overridden by sub-classes: no default implementation is provided.

fr.inria.gforge.telex.application
Class ActionFilter.ExcludeTheseUsersActions

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ActionFilter
      |
      +-fr.inria.gforge.telex.application.ActionFilter.ExcludeTheseUsersActions
```

All Implemented Interfaces:
java.io.Serializable

public static class **ActionFilter.ExcludeTheseUsersActions**
extends ActionFilter

An action filter that excludes all the actions that are issued by a set of users.
Author:
J-M. Busca INRIA/Regal

Fields inherited from class `fr.inria.gforge.telex.application.ActionFilter`

`_document`, `_isActive`, `_name`

Constructor Summary

`public`	`ActionFilter.ExcludeTheseUsersActions(java.lang.String name, java.util.Collection users)` Creates an action filter that excludes all the actions that are issued by the specified set of users.

Method Summary

`boolean`	`isFiltered(Action a)`
`java.lang.String`	`toString()`

Methods inherited from class `fr.inria.gforge.telex.application.ActionFilter`

`getName`, `isActive`, `isFiltered`, `setActive`, `setDocument`, `toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

ActionFilter.ExcludeTheseUsersActions

```
public ActionFilter.ExcludeTheseUsersActions(java.lang.String name,
                                             java.util.Collection users)
```

Creates an action filter that excludes all the actions that are issued by the specified set of users.

(continued from last page)

Parameters:

`name` - the name of the filter.

`users` - the list of users whose actions must be excluded.

Methods

toString

```
public java.lang.String toString()
```

isFiltered

```
public final boolean isFiltered(Action a)
```

Determines whether the specified action must be filtered out. This method is meant to be overridden by sub-classes: no default implementation is provided.

fr.inria.gforge.telex.application
Class ActionFilter.RetainThisDocumentActionsOnly

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ActionFilter
    |
    +-fr.inria.gforge.telex.application.ActionFilter.RetainThisDocumentActionsOnly
```

All Implemented Interfaces:
java.io.Serializable

---

public static class **ActionFilter.RetainThisDocumentActionsOnly**
extends ActionFilter

An action filter that retains only the actions belonging to a document.
Author:
J-M. Busca INRIA/Regal

Fields inherited from class `fr.inria.gforge.telex.application.ActionFilter`

`_document`, `_isActive`, `_name`

Constructor Summary

`public`	`ActionFilter.RetainThisDocumentActionsOnly(java.lang.String name, Document document)` Creates an action filter that retains only the actions belonging to the specified document.

Method Summary

`boolean`	`isFiltered(Action a)`
`java.lang.String`	`toString()`

Methods inherited from class `fr.inria.gforge.telex.application.ActionFilter`

`getName`, `isActive`, `isFiltered`, `setActive`, `setDocument`, `toString`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

ActionFilter.RetainThisDocumentActionsOnly

```
public ActionFilter.RetainThisDocumentActionsOnly(java.lang.String name,
                                                  Document document)
```

Creates an action filter that retains only the actions belonging to the specified document.

(continued from last page)

Parameters:

`name` - the name of the filter.

`document` - the document whose actions must be retained only.

Methods

toString

```
public java.lang.String toString()
```

isFiltered

```
public final boolean isFiltered(Action a)
```

Determines whether the specified action must be filtered out. This method is meant to be overridden by sub-classes: no default implementation is provided.

fr.inria.gforge.telex.application
Class Constraint

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.Constraint
```

All Implemented Interfaces:
java.io.Serializable

---

public class **Constraint**
extends java.lang.Object
implements java.io.Serializable

The generic representation of a constraint in Telex. A constraint expresses a scheduling invariant, defined by its `Constraint.Type`, between two `Action` s. The application adds constraint to a document through the `Document.addFragment(Fragment)`, `Document.addOverlappingFragment(Fragment)` and `ConstraintChecker.getConstraints(Action, Action)` methods.

Most often, a constraint binds two actions of the same document and it is therefore called an *intra-document* constraint. A constraint, however, may bind actions of two distinct documents: such a constraint is called a *cross-documents* constraints, and the corresponding documents are said to be *bound*.

Intra-document constraints. These constraints may bind actions issued by either the same peer or two distinct peers. In general, the application specifies these constraints through the `Document.addFragment(Fragment)` and the `ConstraintChecker.getConstraints(Action, Action)` methods, respectively.

Cross-document constraints. These constraints, together with the actions they bind, must be specified through the `Document.addOverlappingFragment(Fragment)` method, as in the following example:

```
// d1 and d2 are opened document
Action a1 = new Action();
Action a2 = new Action();
Constraint c1 = new Constraint(a1, ENABLES, a2);
Constraint c2 = new Constraint(a2, ENABLES, a1);
Fragment f = new Fragment();
f.addAction(a1);
f.addAction(a2);
f.addConstraint(c1);
f.addConstraint(c2);
// specify where each action goes
a1.setDocument(d1);
a2.setDocument(d2);
d2.addOverlappingFragment(f);
```

A constraint generator may generate constraints in a deterministic fashion or not. A constraint is deterministically generated if the process of generating it only depends on the actions that the constraint binds. In particular, this process does not depend on the site where the constraint is generated, or on the state of the document when the constraint is generated. If a constraint is deterministically generated, Telex saves storage space and network bandwidth consumption by logging the constraint only once.

This class may be sub-classed by Telex applications in order to define application-specific attributes. These attributes will be stored in the multi-log along the generic attributes and they will be transmitted to other peer Telex instances.
Author:
J-M. Busca INRIA/Regal, Abhishek Gupta INRIA/Regal

Nested Class Summary

class	`Constraint.Type` Constraint.Type

Constructor Summary

public	`Constraint()` Creates a constraint object without any type.
public	`Constraint(Action first, Constraint.Type type, Action second)` Creates a constraint of the specified type between the specified actions.

Method Summary

`Constraint`	`copy()` Returns a copy of this constraint.
boolean	`crossesDocuments()` Returns whether this constraint is a cross-document constraint.
boolean	`equals(java.lang.Object object)`
`Action`	`getFirstAction()` Returns the first action that this constraint binds.
`User`	`getIssuer()` Returns the user who issued this constraint.
`Site`	`getLocation()` Returns the site where this constraint was issued.
LogId	`getLogId()` **Deprecated.**
`Peer`	`getPeer()` Returns the peer who issued this constraint.
`Action`	`getSecondAction()` Returns the second action that this constraint binds.
`Constraint.Type`	`getType()` Returns the type of this constraint.
int	`hashCode()`
boolean	`isDeterministic()` Returns the generation mode of this constraint.
boolean	`isLocal()` Returns whether this constraint was issued locally.
void	`read(java.io.DataInput in)` Reads this constraint from the specified input.This method is meant to be overridden by subclasses in order to write their own fields.

void	`setFirstAction(Action action)` **Deprecated.**
void	`setLogId(LogId id)` **Deprecated.**
void	`setSecondAction(Action action)` **Deprecated.**
java.lang.String	`toString()`
static java.lang.String	`toString(Constraint.Type type)`
void	`write(java.io.DataOutput out)` Writes this constraint to the specified output.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

Constraint

```
public Constraint()
```

Creates a constraint object without any type. This mandatory nullary constructor is necessary to read constraints from the multi-log.

Constraint

```
public Constraint(Action first,
                  Constraint.Type type,
                  Action second)
```

Creates a constraint of the specified type between the specified actions. If the type is ENABLES, the constraint specifies that the first action enables the second action. If the type is NOT_BEFORE, the constraint specifies that the first action is not before the second action.

Parameters:

`first` - the first action that this constraint binds.
`type` - the type of the constraint between first and second.
`second` - the second action that this constraint binds.

Methods

toString

```
public java.lang.String toString()
```

(continued from last page)

equals

`public boolean equals(java.lang.Object object)`

hashCode

`public int hashCode()`

getType

`public final Constraint.Type getType()`

Returns the type of this constraint.

Returns:
the type of this constraint.

setFirstAction

`public final void setFirstAction(Action action)`

Deprecated.

Sets the first action of this constraint to the specified action.

For Telex's internal use only.

Parameters:
`action` - the first action of the constraint.

getFirstAction

`public final Action getFirstAction()`

Returns the first action that this constraint binds.

Returns:
the first action that this constraint binds.

setSecondAction

`public final void setSecondAction(Action action)`

Deprecated.

Sets the second action of this constraint to the specified action.

For Telex's internal use only.

Parameters:
`action` - the second action of the constraint.

getSecondAction

`public final Action getSecondAction()`

(continued from last page)

Returns the second action that this constraint binds.

Returns:
the second action that this constraint binds.

setLogId

`public final void setLogId(LogId id)`

Deprecated.

Sets the id of the log this constraint is logged into.

For Telex's internal use only.

Parameters:
`id` - the id of the log this constraint is logged into.

getLogId

`public final LogId getLogId()`

Deprecated.

Returns the id of the log this constraint is logged into.

For Telex's internal use only.

Returns:
the id of the log this constraint is logged into.

getPeer

`public final Peer getPeer()`

Returns the peer who issued this constraint.

This attribute is set after a successful call to the `Document.addFragment(Fragment)` with this constraint as parameter. Before such a call, this attribute is null.

Returns:
the peer who issued this constraint.

getIssuer

`public final User getIssuer()`

Returns the user who issued this constraint.

This attribute is set after a successful call to the `Document.addFragment(Fragment)` with this constraint as parameter. Before such a call, this attribute is null.

Returns:
the user who issued this constraint.

getLocation

`public final Site getLocation()`

(continued from last page)

Returns the site where this constraint was issued.

This attribute is set after a successful call to the `Document.addFragment(Fragment)` with this constraint as parameter. Before such a call, this attribute is null.

Returns:
the site where this constraint was issued.

isLocal

```
public final boolean isLocal()
```

Returns whether this constraint was issued locally.

This attribute is set after a successful call to the `Document.addFragment(Fragment)` with this constraint as parameter. Before such a call, this attribute is null.

Returns:
true if this constraint was issued locally and false otherwise.

isDeterministic

```
public boolean isDeterministic()
```

Returns the generation mode of this constraint. Telex only checks this attribute on constraints issued by a constraint generator.

The default implementation of this method provided by this class always return true. It should be overridden by sub-classes if needed.

Returns:
true if this constraint is generated deterministically, and false otherwise.

crossesDocuments

```
public boolean crossesDocuments()
```

Returns whether this constraint is a cross-document constraint. The method compares actions' document ids. It does not check that corresponding documents are opened and merged at the local site.

This method returns a valid value after a successful call to the `Document.addFragment(Fragment)` with this constraint as parameter. Before such a call, the method throws an IllegalStateException.

Returns:
true if this constraint crosses documents false otherwise.

copy

```
public Constraint copy()
```

Returns a copy of this constraint. The copy is obtained through serialization/deserialization. NOTE that referred-to actions are not copied.

Returns:

write

```
public void write(java.io.DataOutput out)
  throws java.io.IOException
```

(continued from last page)

Writes this constraint to the specified output. This method is meant to be overridden by subclasses in order to write their own fields. The overridden method **must call** this method with super.write() as the **first** statement.

Parameters:

`out` - the output to write this action to.

Throws:

`IOException` - if an I/O error occurs.

read

```
public void read(java.io.DataInput in)
  throws java.io.IOException
```

Reads this constraint from the specified input.This method is meant to be overridden by subclasses in order to write their own fields. The overridden method **must call** this method with super.read() as the **first** statement.

Parameters:

`in` - the input to read this constraint from.

Throws:

`IOException` - if an I/O error occurs.

toString

```
public static java.lang.String toString(Constraint.Type type)
```

fr.inria.gforge.telex.application
Class Constraint.Type

```
java.lang.Object
  |
  +-java.lang.Enum
      |
      +-fr.inria.gforge.telex.application.Constraint.Type
```

All Implemented Interfaces:
java.io.Serializable, java.lang.Comparable

---

public static final class **Constraint.Type**
extends java.lang.Enum

The type of an ACF constraint.

Field Summary

`public static final`	`ENABLES` The *enable* constraint.
`public static final`	`NON_COMMUTING` The *non-commuting* constraint.
`public static final`	`NOT_AFTER` The *not-after* constraint.

Method Summary

`static Constraint.Type`	`valueOf(java.lang.String name)`
`static Constraint.Type[]`	`values()`

Methods inherited from class `java.lang.Enum`

`clone, compareTo, equals, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Fields

(continued from last page)

ENABLES

`public static final fr.inria.gforge.telex.application.Constraint.Type` **`ENABLES`**

The *enable* constraint.

NOT_AFTER

`public static final fr.inria.gforge.telex.application.Constraint.Type` **`NOT_AFTER`**

The *not-after* constraint.

NON_COMMUTING

`public static final fr.inria.gforge.telex.application.Constraint.Type` **`NON_COMMUTING`**

The *non-commuting* constraint.

Methods

values

`public final static Constraint.Type[]` **`values`**`()`

valueOf

`public static Constraint.Type` **`valueOf`**`(java.lang.String name)`

fr.inria.gforge.telex.application
Interface ConstraintChecker

All Known Implementing Classes:
CreationTimeOrder, NonCommutingActions

---

public interface **ConstraintChecker**
extends

An application-specific object that checks for `Constraint`s between `Action`s. Telex associates with each `Document` the constraint checker that the application has registered for the type of the document, if any. Whenever an action a1 is added to the document, Telex checks for constraints between a1 and all existing action a2, whether issued by the same or a distinct peer as a1. Telex performs this check whether action a1 is added by the local peer or a remote peer.

Telex first tests whether actions a1 and a2 are likely to be bound by comparing their keys. If so, Telex then calls the `getConstraints(Action, Action)` method to get the actual set of contraints between a1 and a2. Finally, Telex adds the returned set, if any, to the document. See the the `Constraint.isDeterministic()` method for more details on how theses constraints are logged.

Note that Telex **never** checks for constraints between actions that belong to two distinct documents, even if these documents are bound. See the `Constraint` class for more details on the classification of constraints.

This class provides two examples of constraint checkers.
Author:
P. Sutra INRIA/Regal, J-M. Busca INRIA/Regal

Nested Class Summary

`class`	`ConstraintChecker.CreationTimeOrder` ConstraintChecker.CreationTimeOrder
`class`	`ConstraintChecker.NonCommutingActions` ConstraintChecker.NonCommutingActions

Method Summary

`Fragment`	`getConstraints(Action added, Action existing)` Determines the set of constraints between the specified actions.

Methods

getConstraints

```
public Fragment getConstraints(Action added,
                               Action existing)
```

Determines the set of constraints between the specified actions. Telex calls this method whenever a new action is added to a document. The method is called under specific conditions: see the description of the "telex.scheduling.constraintCheckingMode" and the "telex.scheduling.doCheckActionKeys" properties. If there is no constraint between the specified actions, the method must return an empty fragment.

Parameters:
`added` - a new action added to the document.
`existing` - an existing action of the document.

(continued from last page)

Returns:
the set of constraints between action a and b or an empty fragment if there is no constraint.

fr.inria.gforge.telex.application
Class ConstraintChecker.NonCommutingActions

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ConstraintChecker.NonCommutingActions
```

All Implemented Interfaces:
ConstraintChecker

---

public static final class **ConstraintChecker.NonCommutingActions**
extends java.lang.Object
implements ConstraintChecker

A constraint checker that specifies that no two actions commute.
Author:
J-M. Busca INRIA/Regal

Constructor Summary

`public`	`ConstraintChecker.NonCommutingActions()` Creates constraint generator that specifies that no two actions commute.

Method Summary

`Fragment`	`getConstraints(Action a, Action b)`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `fr.inria.gforge.telex.application.ConstraintChecker`
`getConstraints`

Constructors

ConstraintChecker.NonCommutingActions

```
public ConstraintChecker.NonCommutingActions()
```

Creates constraint generator that specifies that no two actions commute.

Methods

getConstraints

```
public Fragment getConstraints(Action a,
                               Action b)
```

fr.inria.gforge.telex.application
Class ConstraintChecker.CreationTimeOrder

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ConstraintChecker.CreationTimeOrder
```

All Implemented Interfaces:
ConstraintChecker

---

public static final class **ConstraintChecker.CreationTimeOrder**
extends java.lang.Object
implements ConstraintChecker

A constraint checker that forces scheduling in creation-time order. This constraint checker relies on the system clock of cooperating sites, which may not be synchronized with one another.
Author:
J-M. Busca INRIA/Regal

Constructor Summary

`public`	`ConstraintChecker.CreationTimeOrder()` Creates constraint generator that that forces scheduling in creation-time order.

Method Summary

`Fragment`	`getConstraints(Action a, Action b)`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `fr.inria.gforge.telex.application.ConstraintChecker`

`getConstraints`

Constructors

ConstraintChecker.CreationTimeOrder

```
public ConstraintChecker.CreationTimeOrder()
```

Creates constraint generator that that forces scheduling in creation-time order.

Methods

getConstraints

```
public Fragment getConstraints(Action a,
                               Action b)
```

(continued from last page)

fr.inria.gforge.telex.application
Interface DocumentState

public interface **DocumentState**
extends java.io.Serializable

The state of a `Document` as maintained by a `TelexApplication`. This may be the Java object representing the document itself or any object that the application uses to identify such object. Document states are used to specify `Schedule`s or materialized `StateSnapshot`s.

When storing a document state on persistent storage, Telex optimizes disk utilization by using a content-based naming scheme, as follows. Telex splits the state into fixed-size fragments, computes the SHA-1 hash of each fragment, and stores it in a file named after the hash value. Thus, state fragments that remain unchanged from one snapshot to another are reused instead of being stored twice. Moreover, Telex stores fragments in a space common to all users, thus enabling fragment sharing between users.

The application may help Telex to improve fragment re-use and sharing by specifying the appropriate set of fragments to use. This is the purpose of the `split()` and `assemble(Serializable[])` methods, which Telex calls before saving and after restoring the document state, respectively.

The state of a document must be serializable in order to be stored on persistent storage. In the current implementation, Telex also use serialization - deserialization to generate copies of the state.
Author:
J-M. Busca INRIA/Regal

Field Summary

`public static final`	`DEFAULT_FRAGMENT_SIZE` The default size of state fragments. Value: **`8192`**

Method Summary

`void`	`assemble(java.io.Serializable[] parts)` Provides the fragments that make up this document state.
`java.io.Serializable[]`	`split()` Returns the fragments that make up this document state.

Fields

DEFAULT_FRAGMENT_SIZE

`public static final int` **`DEFAULT_FRAGMENT_SIZE`**

The default size of state fragments.
Constant value: **`8192`**

Methods

split

`public java.io.Serializable[]` **`split`**`()`

(continued from last page)

Returns the fragments that make up this document state. Telex calls this method before storing this state on persistent storage. It then stores each returned fragment in file named after the hash of the fragment. If the method returns null, Telex splits this state in fragments of size `DEFAULT_FRAGMENT_SIZE`.

Returns:
the fragments that make up this document state.

assemble

```
public void assemble(java.io.Serializable[] parts)
```

Provides the fragments that make up this document state. If the `split()` returned a non-null value, Telex calls this method after loading document state from persistent storage.

Parameters:
`parts` - the fragments that make up this state.

fr.inria.gforge.telex.application
Class Fragment

```
java.lang.Object
   |
   +-fr.inria.gforge.telex.application.Fragment
```

All Implemented Interfaces:
java.io.Serializable

public class **Fragment**
extends java.lang.Object
implements java.io.Serializable

A set of `Action`s and `Constraint`s considered as a whole. See the `Document.addFragment(Fragment)` method.
Author:
J-M. Busca INRIA/Regal

Field Summary

`public`	`_id`

Constructor Summary

`public`	`Fragment()` Creates an empty fragment.
`public`	`Fragment(java.util.HashSet actions, java.util.HashSet constraints)` Creates a new fragment containing the specified sets of actions and constraints.
`public`	`Fragment(java.util.Collection actions, java.util.Collection constraints)` Creates a new fragment containing the specified collections of actions and constraints.
`public`	`Fragment(Fragment fragment)` Creates a new fragment which is copy of the specified fragment.

Method Summary

`boolean`	`add(Action action)` Adds the specified action to this fragment.
`boolean`	`add(Constraint constraint)` Adds the specified constraint to this fragment.
`boolean`	`add(Fragment fragment)` Adds the elements of the specified fragment to this fragment.
`void`	`clear()` Removes all of the elements of this fragment.
`boolean`	`contains(Fragment f)`

`Fragment`	`darkside()`
`java.util.Set`	`getActions()` Returns the set of actions of this fragment.
`java.util.Set`	`getConstraints()` Returns the set of constraints of this fragment.
`java.util.Set`	`getDocuments()`
`boolean`	`isClosed()`
`boolean`	`isEmpty()` Checks whether this fragment is empty.
`boolean`	`iSoundWith(Fragment candidate)`
`Fragment`	`projectOn(java.util.Set docs)`
`void`	`reBuild()` **Deprecated.**
`boolean`	`remove(Fragment fragment)` Removes the elements of the specified fragment from this fragment.
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

_id

```
public long _id
```

Constructors

Fragment

```
public Fragment()
```

Creates an empty fragment.

Fragment

```
public Fragment(java.util.HashSet actions,
                java.util.HashSet constraints)
```

(continued from last page)

Creates a new fragment containing the specified sets of actions and constraints. The fragment is backed by the specified sets: changes to the sets are reflected in the fragment and vice-versa.

Parameters:

`actions` - the set of actions to initially place in this fragment, or null if the set is empty.
`constraints` - the set of constraints to initially place in this fragment, or null if the set is empty.

Fragment

```
public Fragment(java.util.Collection actions,
                java.util.Collection constraints)
```

Creates a new fragment containing the specified collections of actions and constraints. The constructor creates a copy of the specified collections: changes to the collections are not reflected in the fragment and vice-versa. This is a convenience method which is equivalent to:

```
Fragment fragment = new Fragment();
for (Action a : actions) {
        fragment.add(a);
}
for (Constraint c : constraints) {
        fragment.add(c);
}
```

Parameters:

`actions` - the collection of actions to initially place in this fragment, or null if the collection is empty.
`constraints` - the collection of constraints to initially place in this fragment, or null if the collection is empty.

Fragment

```
public Fragment(Fragment fragment)
```

Creates a new fragment which is copy of the specified fragment.

Parameters:

`fragment` - the fragment to duplicate.

Methods

toString

```
public java.lang.String toString()
```

getActions

```
public java.util.Set getActions()
```

Returns the set of actions of this fragment. The returned set is backed by this fragment, so changes to this fragment are reflected in the set, and vice-versa.

Returns:

(continued from last page)

the set of actions of this fragment.

getConstraints

`public java.util.Set getConstraints()`

Returns the set of constraints of this fragment. The returned set is backed by this fragment, so changes to this fragment are reflected in the set, and vice-versa.

Returns:
the set of constraints of this fragment.

isEmpty

`public boolean isEmpty()`

Checks whether this fragment is empty. A fragment is empty if it contains no action and no constraint.

Returns:
true if this fragment is empty and false otherwise.

add

`public boolean add(Action action)`

Adds the specified action to this fragment.

Parameters:
`action` - the action to add.

Returns:
true if this fragment did not already contain the specified action, and false otherwise.

add

`public boolean add(Constraint constraint)`

Adds the specified constraint to this fragment.

Parameters:
`constraint` - the constraint to add.

Returns:
true if this fragment did not already contain the specified constraint, and false otherwise.

add

`public boolean add(Fragment fragment)`

Adds the elements of the specified fragment to this fragment.

Parameters:
`fragment` - the fragment whose elements are to be added.

Returns:
true if this fragment changed as a result of the call, and false otherwise.

(continued from last page)

remove

`public boolean **remove**(Fragment fragment)`

Removes the elements of the specified fragment from this fragment.

Parameters:
`fragment` - the fragment whose elements are to be removed.

Returns:
true if this fragment changed as a result of the call, and false otherwise.

---

reBuild

`public void **reBuild**()`

Deprecated.

Rebuilds this fragment. Telex calls this method after actions have been logged, so as to take their id being set into account.

For Telex's internal use only.

---

clear

`public void **clear**()`

Removes all of the elements of this fragment. This fragment will be empty after this call returns.

---

isClosed

`public boolean **isClosed**()`

---

iSoundWith

`public boolean **iSoundWith**(Fragment candidate)`

---

projectOn

`public Fragment **projectOn**(java.util.Set docs)`

---

contains

`public boolean **contains**(Fragment f)`

---

getDocuments

`public java.util.Set **getDocuments**()`

(continued from last page)

darkside

```
public Fragment darkside()
```

fr.inria.gforge.telex.application
Class ProcessingParameters

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.ProcessingParameters
```

public class **ProcessingParameters**
extends java.lang.Object

The application-defined parameters for processing a `Document`. These parameters fall into the following categories:

- Multi-log reading: the *application action* [resp. *application constraint*] field specifies the application-specific class of actions that make up the document. Telex instantiates this class when reading the multi-log of the document. This class must have a public nullary constructor. Default value: the `Action` [resp. `Constraint`] class.
- State storing: the *initial state* field specifies the `DocumentState` at creation time. This is the state of the first `Schedule`s returned to the application. Default value: no initial state. The *fragment size* field specifies the optimal size of the state fragments when storing a materialized `StateSnapshot` on persistent storage. Default value: `DocumentState.DEFAULT_FRAGMENT_SIZE`.
- Constraint checking: the *constraint checker* field specifies the `ConstraintChecker` that Telex must use for the document. Default value: no constraint checker.
- Action scheduling: the *scheduling parameters* field specifies the `SchedulingParameters` to use for the document. These parameters can be changed later on on a per-document basis. Default value: `SchedulingParameters.DEFAULT_PARAMETERS`.
- Functional extensions: the *replica reconciler* [resp. *schedule generator*] field specify the application-specific `fr.inria.gforge.telex.reconciliation.ReplicaReconciler` [resp. `ScheduleGenerator`] class to use for the document. Telex instantiates this class for each open document. This class must have a public nullary constructor. Default value: the `fr.inria.gforge.telex.reconciliation.ReplicaReconciler` [resp. `ScheduleGenerator`] class.

Most `TelexApplication`s only have to define the initial state of a document, the actions and constraints it is made of and a specific constraint checker. Some applications may need to additionally define specific default scheduling parameters and state fragment size. Telex provides default implementations of the replica reconciler and the schedule generator that suit most needs.

Only applications with very specific requirements need to provide their own replica reconciler and/or schedule generator.
Author:
J-M. Busca INRIA/Regal

Field Summary

`public static final`	`DEFAULT_PARAMETERS` The constant for "default processing parameters".

Constructor Summary

`public`	`ProcessingParameters()` Creates an instance of processing parameters with default values.
`public`	`ProcessingParameters(DocumentState state, java.lang.Class action, java.lang.Class constraint, ConstraintChecker checker)` Creates an instance of processing parameters with the specified initial state, application action, application constraint and constraint checker.

Method Summary

`java.lang.Class`	`getApplicationAction()` Returns the application action field of these parameters.

java.lang.Class	getApplicationConstraint() Returns the application constraint field of these parameters.
ConstraintChecker	getConstraintChecker() Returns the constraint checker field of these parameters.
int	getFragmentSize() Return the fragment size field of these parameters.
DocumentState	getInitialState() Returns the initial state field of these parameters.
java.lang.Class	getReplicaReconciler() Returns the replica reconciler field of these parameters.
java.lang.Class	getScheduleGenerator() Returns the schedule generator field of these parameters.
SchedulingParameters	getSchedulingParameters() Returns the default scheduling parameters field of these parameters.
Constraint	instanctiateConstraint() Returns a new instance of the constraint field of these parameters.
Action	instantiateAction() Returns a new instance of the action field of these parameters.
Action	instantiateApplicationAction() Returns a new instance of the application action field of these parameters.
Constraint	instantiateApplicationConstraint() Returns a new instance of the application constraint field of these parameters.
DocumentState	instantiateDocumentState() Returns a new instance of the initial document state field of these parameters.
ReplicaReconciler	instantiateReplicaReconciler(DocumentImpl document) Returns a new instance of the replica reconciler.
ScheduleGenerator	instantiateScheduleGenerator(SchedulingContext context) Returns a new instance of the schedule generator field of these parameters.
void	setApplicationAction(java.lang.Class action) Sets the application action field of these parameters.
void	setApplicationConstraint(java.lang.Class constraint) Sets the application constraint field of these parameters.
void	setConstraintChecker(ConstraintChecker checker) Sets the constraint checker field of these parameters.
void	setFragmentSize(int size) Sets the fragment size field of these parameters.
void	setInitialState(DocumentState state) Sets the initial state of these parameters.
void	setReplicaReconciler(java.lang.Class reconciler) Sets the replica reconciler field of these parameters.

`void`	`setScheduleGenerator(java.lang.Class generator)` Sets the schedule generator field of these parameters.
`void`	`setSchedulingParameters(SchedulingParameters parameters)` Sets the scheduling parameters field of these parameters.
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

DEFAULT_PARAMETERS

```
public static final fr.inria.gforge.telex.application.ProcessingParameters
DEFAULT_PARAMETERS
```

The constant for "default processing parameters". This constant is created by calling the `ProcessingParameters()` constructor.

Constructors

ProcessingParameters

```
public ProcessingParameters()
```

Creates an instance of processing parameters with default values.

ProcessingParameters

```
public ProcessingParameters(DocumentState state,
                            java.lang.Class action,
                            java.lang.Class constraint,
                            ConstraintChecker checker)
```

Creates an instance of processing parameters with the specified initial state, application action, application constraint and constraint checker. Others fields are set to default values.

The object passed as the initial state parameter **must never** be updated after the constructor returns. The constructor checks that the application action and application constraint classes can actually be instantiated with a nullary constructor.

Parameters:

`state` - the initial state field.
`action` - the application action field.
`constraint` - the application constraint field.
`checker` - the constraint checker field.

Throws:

`UninstantiableClassException` - if the action or constraint parameter cannot be instantiated with a nullary constructor.

Methods

(continued from last page)

toString

`public java.lang.String **toString**()`

setApplicationAction

`public void **setApplicationAction**(java.lang.Class action)`

Sets the application action field of these parameters. This field is a class that extends the Action class. It must have a nullary constructor.

Parameters:
`action` - the application action to set.

getApplicationAction

`public java.lang.Class **getApplicationAction**()`

Returns the application action field of these parameters.

Returns:
the application action field of these parameters.

setApplicationConstraint

`public void **setApplicationConstraint**(java.lang.Class constraint)`

Sets the application constraint field of these parameters. This field is a class that extends the Constraint class. It must have a nullary constructor.

Parameters:
`constraint` - the application constraint to set.

getApplicationConstraint

`public java.lang.Class **getApplicationConstraint**()`

Returns the application constraint field of these parameters.

Returns:
the application constraint field of these parameters.

instantiateApplicationAction

`public Action **instantiateApplicationAction**()`

Returns a new instance of the application action field of these parameters.

Returns:
a new instance of the application action field of these parameters.

instantiateApplicationConstraint

`public Constraint **instantiateApplicationConstraint**()`

Returns a new instance of the application constraint field of these parameters.

(continued from last page)

Returns:
a new instance of the application constraint field of these parameters.

setInitialState

`public void setInitialState(DocumentState state)`

Sets the initial state of these parameters. Note that the object passed as parameter **must never** be updated after this method returns.

Parameters:
`state` - the initial state to set.

getInitialState

`public DocumentState getInitialState()`

Returns the initial state field of these parameters.

Returns:
the constraint checker field of these parameters.

setFragmentSize

`public void setFragmentSize(int size)`

Sets the fragment size field of these parameters.

Parameters:
`size` - the fragment size to set.

getFragmentSize

`public int getFragmentSize()`

Return the fragment size field of these parameters.

Returns:
the fragment size field of these parameters.

setConstraintChecker

`public void setConstraintChecker(ConstraintChecker checker)`

Sets the constraint checker field of these parameters.

Parameters:
`checker` - the constraint checker to set.

getConstraintChecker

`public ConstraintChecker getConstraintChecker()`

Returns the constraint checker field of these parameters.

Returns:
the constraint checker field of these parameters.

setSchedulingParameters

```
public void setSchedulingParameters(SchedulingParameters parameters)
```

Sets the scheduling parameters field of these parameters.

Parameters:
`parameters` - the scheduling parameters to set.

getSchedulingParameters

```
public SchedulingParameters getSchedulingParameters()
```

Returns the default scheduling parameters field of these parameters.

Returns:
the default scheduling parameters field of these parameters.

setReplicaReconciler

```
public void setReplicaReconciler(java.lang.Class reconciler)
  throws UninstantiableClassException
```

Sets the replica reconciler field of these parameters. This field is a class that extends the ReplicaReconciler class. It must have a nullary constructor.

Parameters:
`reconciler` - the replica reconciler to set.

Throws:
`UninstantiableClassException` - if the replica reconciler cannot be instantiated with a nullary constructor.

getReplicaReconciler

```
public java.lang.Class getReplicaReconciler()
```

Returns the replica reconciler field of these parameters.

Returns:
the replica reconciler field of these parameters.

setScheduleGenerator

```
public void setScheduleGenerator(java.lang.Class generator)
  throws UninstantiableClassException
```

Sets the schedule generator field of these parameters. This field is a class that extends the ScheduleGenerator class. It must have a nullary constructor.

Parameters:
`generator` - the schedule generator to set.

Throws:
`UninstantiableClassException` - if the generator cannot be instantiated with a nullary constructor.

getScheduleGenerator

```
public java.lang.Class getScheduleGenerator()
```

(continued from last page)

Returns the schedule generator field of these parameters.

Returns:
the schedule generator field of these parameters.

instantiateAction

`public Action instantiateAction()`

Returns a new instance of the action field of these parameters.

Returns:
a new instance of the action field of these parameters.

instanctiateConstraint

`public Constraint instanctiateConstraint()`

Returns a new instance of the constraint field of these parameters.

Returns:
a new instance of the constraint field of these parameters.

instantiateDocumentState

`public DocumentState instantiateDocumentState()`

Returns a new instance of the initial document state field of these parameters.

Returns:
a new instance of the initial document state field of these parameters.

instantiateScheduleGenerator

`public ScheduleGenerator instantiateScheduleGenerator(SchedulingContext context)`

Returns a new instance of the schedule generator field of these parameters.

Parameters:
`context` - the scheduling context to pass to the constructor.

Returns:
a new instance of the schedule generator field of these parameters.

instantiateReplicaReconciler

`public ReplicaReconciler instantiateReplicaReconciler(DocumentImpl document)`

Returns a new instance of the replica reconciler. The method assumes the following values, in order: (i) the "telex.implementation.reconcilerClass" telex property; (ii) the replica reconciler field of these processing parameters.

Parameters:
`document` - the document to pass to the constructor.

Returns:
a new instance of the replica reconciler.

fr.inria.gforge.telex.application
Class SchedulingParameters

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.SchedulingParameters
```

public class **SchedulingParameters**
extends java.lang.Object

The action scheduling parameters of a `Document`. These parameters control how often Telex automatically calls the `TelexApplication.execute(Document, ScheduleGenerator)` method in response to new fragments being added to the document. Telex calls this method whenever one of the following conditions is true:

- the number of fragments added to the document since the last call exceeds a given value (threshold parameter),
- the time since a new fragment was added after the last call exceeds a given value (delay parameter).

When evaluating these conditions, Telex considers fragments added by both local site and remote sites, as well as calls to execute() that the application may trigger unconditionally by calling the `Document.executeNow(boolean)` method.

Automatic calls to execute() may be de-activated by specifying the UNSPECIFIED_THRESHOLD and UNSPECIFIED_DELAY values for the threshold and delay parameters, respectively. In this case, the application may obtain new sound schedules only by calling executeNow().

When a document is opened, it is assigned the default scheduling parameters defined for its type. These parameters may be dynamically changed by calling the `Document.setSchedulingParameters(SchedulingParameters)`. When several documents are bound by cross-document constraints, Telex automatically set the scheduling parameters of all of the bound documents to the most restrictive values currently defined for these documents.

Author:
J-M. Busca INRIA/Regal

Field Summary

`public static final`	`DEFAULT_PARAMETERS` The constant for "default scheduling parameters".
`public static final`	`NO_DELAY` The constant for "no delay". Value: **`0`**
`public static final`	`NO_THRESHOLD` The constant for "no threshold". Value: **`1`**
`public static final`	`UNSPECIFIED_DELAY` The constant for "unspecified delay". Value: **`-1`**
`public static final`	`UNSPECIFIED_THRESHOLD` The constant for "unspecified threshold". Value: **`0`**

Constructor Summary

public	`SchedulingParameters()` Creates an instance of scheduling parameters with default values.
public	`SchedulingParameters(int threshold, long delay)` Creates an instance of scheduling parameters with the specified values.
public	`SchedulingParameters(SchedulingParameters parameters)` Creates an instance of scheduling parameters by duplicating the specified values.

Method Summary

long	`getDelay()` Returns the delay value of these scheduling parameters.
int	`getThreshold()` Returns the threshold value of these scheduling parameters.
void	`merge(SchedulingParameters parameters)` Merges the specified scheduling parameters into these scheduling parameters.
java.lang.String	`toString()`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

DEFAULT_PARAMETERS

`public static final fr.inria.gforge.telex.application.SchedulingParameters` **`DEFAULT_PARAMETERS`**

The constant for "default scheduling parameters". This constant is created by calling the `SchedulingParameters()` constructor.

NO_THRESHOLD

`public static final int` **`NO_THRESHOLD`**

The constant for "no threshold". If threshold is set to this value, sound schedules are computed every time a new fragment is added to the document, regardless of the delay value.
Constant value: **`1`**

UNSPECIFIED_THRESHOLD

`public static final int` **`UNSPECIFIED_THRESHOLD`**

The constant for "unspecified threshold". If threshold is set to this value, schedule computation is only governed by the delay parameter.
Constant value: **`0`**

NO_DELAY

`public static final int` **`NO_DELAY`**

(continued from last page)

The constant for "no delay". If delay is set to this value, sound schedules are computed every time a new fragment is added to the document, regardless of the threshold value.
Constant value: **0**

UNSPECIFIED_DELAY

`public static final int` **`UNSPECIFIED_DELAY`**

The constant for "unspecified delay". If delay is set to this value, schedule computation is only governed by the threshold parameter.
Constant value: **-1**

Constructors

SchedulingParameters

`public` **`SchedulingParameters`**`()`

Creates an instance of scheduling parameters with default values. The threshold parameter is set to NO_THRESHOLD, and the delay parameter is set to NO_DELAY.

SchedulingParameters

```
public SchedulingParameters(int threshold,
                            long delay)
```

Creates an instance of scheduling parameters with the specified values. if the threshold (resp. delay) value is less or equal to 0, it is silently set to NO_THRESHOLD (resp. NO_DELAY).

Parameters:
- `threshold` - the value of the threshold parameter.
- `delay` - the value of the delay parameter, in millisecond.

SchedulingParameters

`public` **`SchedulingParameters`**`(SchedulingParameters parameters)`

Creates an instance of scheduling parameters by duplicating the specified values.

Parameters:
- `parameters` - the parameters to duplicate.

Methods

getThreshold

`public int` **`getThreshold`**`()`

Returns the threshold value of these scheduling parameters.

Returns:
the threshold value of these scheduling parameters.

getDelay

`public long` **`getDelay`**`()`

Returns the delay value of these scheduling parameters.

(continued from last page)

Returns:
the delay value of these scheduling parameters.

toString

`public java.lang.String **toString**()`

merge

`public void **merge**(SchedulingParameters parameters)`

Merges the specified scheduling parameters into these scheduling parameters. This method compares both parameters and retains the most restrictive, i.e. the smallest, value of each of the threshold and delay fields.

Parameters:
`parameters` - the scheduling parameters to merge into these ones.

fr.inria.gforge.telex.application
Class StateSnapshot

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.application.StateSnapshot
```

All Implemented Interfaces:
java.lang.Comparable, java.io.Serializable

---

public class **StateSnapshot**
extends java.lang.Object
implements java.io.Serializable, java.lang.Comparable

A snapshot of the state of a `Document`. It is defined by the `Schedule` of actions producing the state that the snapshot represents and is identified by a name. The `Document.defineSnapshot(StateSnapshot)` uses this name to save the Snapshot to persistent storage and also records the time when the snapshot is saved.

A snapshot is said to be *materialized* if the `DocumentState` it corresponds to is provided when the snapshot is created. A snapshot is said to be *committed* if all of the actions of the corresponding schedule are committed. Committed snapshots are the only snapshots that may be considered as garbage-collection points.

This class may be sub-classed by applications, for instance to provide the description the snapshot. A snapshot must be serializable in order to be saved as part of the persistent state of the document.
Author:
J-M. Busca INRIA/Regal

Constructor Summary

`public`	`StateSnapshot(java.lang.String name, Schedule schedule)` Creates a simple snapshot with the specified name corresponding to the specified schedule.
`public`	`StateSnapshot(java.lang.String name, Schedule schedule, DocumentState state)` Creates a materialized snapshot with the specified name corresponding to the specified schedule and state.
`public`	`StateSnapshot(DocumentImpl document)` **Deprecated.**

Method Summary

`int`	`compareTo(StateSnapshot other)` **Deprecated.**
`boolean`	`equals(java.lang.Object object)` **Deprecated.**
`Document`	`getDocument()` Returns the document that this snapshot relates to.
`java.lang.String`	`getName()` Returns the name of this snapshot.
`Schedule`	`getSchedule()` Returns the schedule of actions producing the state that this snapshot defines.

`long`	`getSize()` Returns the size of this snapshot on disk.
`DocumentState`	`getState()` Returns the state that this snapshot defines.
`long`	`getTime()` Returns the time this snapshot was saved on disk.
`int`	`hashCode()` **Deprecated.**
`boolean`	`isCommitted()` Determines whether this snapshot is committed.
`boolean`	`isMaterialized()` Determines whether this snapshot is materialized.
`void`	`load(java.lang.String pathname)` **Deprecated.**
`void`	`remove()` **Deprecated.**
`void`	`store(java.lang.String pathname)` **Deprecated.**
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Comparable`

`compareTo`

Constructors

StateSnapshot

```
public StateSnapshot(java.lang.String name,
                     Schedule schedule)
```

Creates a simple snapshot with the specified name corresponding to the specified schedule.

Parameters:

`name` - the name of the snapshot.
`schedule` - the schedule that defines this snapshot.

StateSnapshot

```
public StateSnapshot(java.lang.String name,
                     Schedule schedule,
                     DocumentState state)
```

(continued from last page)

Creates a materialized snapshot with the specified name corresponding to the specified schedule and state. The constructor saves a copy of the specified state object, which must not be updated during the time the constructor executes.

Parameters:
`name` - the name of the snapshot.
`schedule` - the schedule that defines this snapshot.
`state` - the document state that the execution of the schedule yields.

StateSnapshot

`public StateSnapshot(DocumentImpl document)`

Deprecated.

Creates a void snapshot associated with the specified document. The snapshot must be subsequently loaded from persistent storage by calling the `load(String)` method.

This constructor is for Telex's internal use only.

Parameters:
`document` - the document to associate the snapshot with.

Methods

toString

`public java.lang.String toString()`

getName

`public final java.lang.String getName()`

Returns the name of this snapshot.

Returns:
the name of this snapshot.

getTime

`public final long getTime()`

Returns the time this snapshot was saved on disk. This attribute is set by Telex when the `Document.defineSnapshot(StateSnapshot)` method is called on this snapshot.

Returns:
the time this snapshot was created.

getSize

`public final long getSize()`

Returns the size of this snapshot on disk. This attribute is set by Telex when the `Document.defineSnapshot(StateSnapshot)` method is called on this snapshot. This method returns the total size of the snapshot, regardless of state fragment being shared.

Returns:
the tsize of this snapshot on disk.

getSchedule

```
public final Schedule getSchedule()
```

Returns the schedule of actions producing the state that this snapshot defines.

Returns:
the schedule of actions producing the state that this snapshot defines.

getState

```
public final DocumentState getState()
  throws java.io.IOException
```

Returns the state that this snapshot defines. If this snapshot is not materialized, this method returns null.

Note that when a snapshot is retrieved from persistent storage, the corresponding state, if any, is not loaded into memory until this method is called. In order to know whether this snapshot is materialized, call the `isMaterialized()` method instead.

Returns:
the state that this snapshot defines, or null if this snapshot is not materialized.

Throws:
`IOException` - if an I/O error occurs.

getDocument

```
public final Document getDocument()
```

Returns the document that this snapshot relates to. It is a convenience method equivalent to:

```
getSchedule().getDocument();
```

Returns:
the document that this snapshot relates to.

compareTo

```
public int compareTo(StateSnapshot other)
```

Deprecated.

equals

```
public boolean equals(java.lang.Object object)
```

Deprecated.

hashCode

```
public int hashCode()
```

Deprecated.

isMaterialized

```
public final boolean isMaterialized()
```

Determines whether this snapshot is materialized. This method is semantically equivalent to:

```
getState() != null
```

Nonetheless, this method should be preferred to `getState()` since calling the latter method actually load the state in memory.

Returns:
true if this snapshot is materialized, and false otherwise.

isCommitted

```
public final boolean isCommitted()
```

Determines whether this snapshot is committed.

Returns:
true if this snapshot is committed, and false otherwise.

store

```
public void store(java.lang.String pathname)
  throws java.io.IOException
```

Deprecated.

Stores this snapshot in the file with the specified pathname. The method stores the associated schedule and the associated materialized state, if any.

This method is for Telex's internal use only.

Parameters:
`pathname` - the pathname of the file to store this snapshot to.

Throws:
`IOException` - if an I/O error occurs.

load

```
public void load(java.lang.String pathname)
  throws java.io.IOException
```

Deprecated.

(continued from last page)

Loads this snapshot from the file with the specified pathname. The method loads the associated schedule. The materialized state, if any, will be loaded when the `getState()` method is called.

Parameters:
`pathname` - the pathname of the file to load this snapshot from.

Throws:
`IOException` - if an I/O error occurs.

---

remove

```
public void remove()
  throws java.io.IOException
```

Deprecated.

Removes this snapshot from persistent storage. The method removes the associated schedule and the associated materialized state, if any.

This method is for Telex's internal use only.

Throws:
`IOException` - if an I/O error occurs.

fr.inria.gforge.telex.application
Interface TelexApplication

public interface **TelexApplication**
extends

An application that uses the services of `Telex`. This interface defines upcalls that Telex uses to notify the application of various events.

All these events relate to a specific `Document` that the application has opened by calling the `Telex.openDocument(String, Telex.OpenMode)` method. Telex never notifies the application of events regarding documents that the application has not opened explicitely, or that it has closed. Some of these events relate to bound documents (see `Constraint`).

Author:
P. Sutra INRIA/Regal, J-M. Busca INRIA/Regal

Method Summary

`void`	`bindDocument(Document opened, Document bound)` Notifies this application that the specified documents are bound.
`void`	`execute(Document document, Schedule schedule)` Requests this application to execute the specified schedule on the specified document.
`void`	`execute(Document document, ScheduleGenerator generator)` Requests this application to activate the specified schedule generator on the specified document.

Methods

execute

```
public void execute(Document document,
        ScheduleGenerator generator)
```

Requests this application to activate the specified schedule generator on the specified document. Telex automatically calls this method whenever conditions for generating new schedules are met on the specified document (see `SchedulingParameters`). This method is also called as a result of this application calling `Document.executeNow(boolean)` on the specified document.

Parameters:
`document` - the document generator relate to.
`generator` - the schedule generator for document.

bindDocument

```
public void bindDocument(Document opened,
        Document bound)
```

Notifies this application that the specified documents are bound. Telex calls this method when it notices that document opened, currently opened by this application, is bound with document bound. For each pair (opened, bound), Telex calls this method only once after document opened has been opened.

Note that (i) when this method is called, document bound may be opened or closed (ii) if opened, the application that edits it may be this application or another application.

(continued from last page)

Parameters:

`opened` - a document currently opened by this application.

`bound` - the document that is bound to current.

execute

```
public void execute(Document document,
          Schedule schedule)
```

Requests this application to execute the specified schedule on the specified document. This method is only called when the specified document is bound to another document opened by some application app. Application app uses it to synchronize with this application in order to display related schedules on the bound documents.

Parameters:

`document` - the document schedule relates to.

`schedule` - the schedule to apply on document.

Package fr.inria.gforge.telex.extensions

Provides classes and interfaces for extending the core functionalities of Telex.

fr.inria.gforge.telex.extensions
Class ComponentDescriptor

```
java.lang.Object
   |
   +-fr.inria.gforge.telex.extensions.ComponentDescriptor
```

All Implemented Interfaces:
java.lang.Iterable

---

public final class **ComponentDescriptor**
extends java.lang.Object
implements java.lang.Iterable

The descriptor of a connected component of an action-constraint graph. It is used by the `ComponentIterableScheduleGenerator` to describe each component of the graph for the application. The returned information includes (i) the id of the component, (ii) whether the component has been updated since last schedule generation and (iii) an iterator over sound schedules that can be drawn from the component.

Note that *independent* actions, e.g. actions that are not bound by any constraint, are grouped into a single fictitious component. By convention, this component is identified by the `INDEPENDENT_COMPONENT` id, it is marked as updated and if the component is empty, the iterator returns a schedule without any action.

Author:
J-M. Busca INRIA/Regal

Field Summary

`public static final`	`INDEPENDENT_COMPONENT` Value:

Method Summary

`java.lang.String`	`getId()` Returns the id of this component.
`boolean`	`isUpdated()` Returns whether this component was updated.
`java.util.Iterator`	`iterator()` Returns an iterator over the schedules of this component.
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Iterable`

`iterator`

(continued from last page)

Fields

INDEPENDENT_COMPONENT

`public static final java.lang.String` **`INDEPENDENT_COMPONENT`**

Constant value:

Methods

toString

`public java.lang.String` **`toString`**`()`

---

getId

`public java.lang.String` **`getId`**`()`

Returns the id of this component.

Returns:
the id of this component.

---

isUpdated

`public boolean` **`isUpdated`**`()`

Returns whether this component was updated.

Returns:
whether this component was updated.

---

iterator

`public java.util.Iterator` **`iterator`**`()`

Returns an iterator over the schedules of this component.

Returns:
an iterator over the schedules of this component.

fr.inria.gforge.telex.extensions
Class ComponentIterableScheduleGenerator

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.extensions.ScheduleGenerator
      |
      +-fr.inria.gforge.telex.extensions.ComponentIterableScheduleGenerator
```

public class **ComponentIterableScheduleGenerator**
extends ScheduleGenerator

A connected component-aware iterable `ScheduleGenerator`. The purpose of this generator is to enhance both user's experience and scheduling performance by handling the connected components of an action-constraint graph independently. Each component is described by a `ComponentDescriptor` that returns the set of schedules for this component.

The design of this generator is similar to that of the `IterableScheduleGenerator` class. Telex allocates a primary component-aware generator per action-constraint graph. This generator in turn creates a primary iterable generator responsible for each component of the graph. When called to allocate a secondary component-aware generator for a given document, the primary creates a secondary iterable generator for this document on each component. Note that it does so even when the component does not contain actions of the document. Note also that to ease application programming, independent actions are grouped into a fictitious graph component.
Author:
J-M. Busca INRIA/Regal

Fields inherited from class `fr.inria.gforge.telex.extensions.ScheduleGenerator`
`_schedulingContext`, `DEFAULT_SCHEDULER`

Constructor Summary

`public`	`ComponentIterableScheduleGenerator(SchedulingContext context)` Creates a primary component-aware iterable schedule generator on the specified scheduling context.

Method Summary

`Schedule`	`getCommittedSchedule()` Returns the schedule that has been committed so far.
`ComponentDescriptor[]`	`getComponentDescriptors()` Returns the set of descriptors of the connected components.
`Action[]`	`getDeadActions()` Returns the set of dead actions.
`ScheduleGenerator`	`getInstance(DocumentImpl document)`
`java.lang.String`	`toString()`
`boolean`	`usesSplitGraph()`

Methods inherited from class `fr.inria.gforge.telex.extensions.ScheduleGenerator`

`getBaseStates, getBoundDocuments, getInstance, getParentScheduler, getStateGraph, isReleased, release, usesSplitGraph`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

ComponentIterableScheduleGenerator

`public **ComponentIterableScheduleGenerator**(SchedulingContext context)`

Creates a primary component-aware iterable schedule generator on the specified scheduling context.

Parameters:
`context` - the scheduling context to process.

Methods

toString

`public java.lang.String **toString**()`

getInstance

`public ScheduleGenerator **getInstance**(DocumentImpl document)`

Returns a schedule generator instance responsible for the specified document. Telex calls this method for each of the bound documents the graph to process relates to, even if it contains only one element. Telex passes the returned schedule generator to the execute() method.

usesSplitGraph

`public boolean **usesSplitGraph**()`

Returns whether this generator uses a split graph. The `fr.inria.gforge.telex.scheduling.Scheduler` calls this method to allocate an action-constraint graph of the appropriate type. The default implementation of this method returns false.

getDeadActions

`public Action[] **getDeadActions**()`

Returns the set of dead actions.

Returns:
the set of dead actions.

getCommittedSchedule

`public Schedule **getCommittedSchedule**()`

(continued from last page)

Returns the schedule that has been committed so far.

Returns:
the schedule that has been committed so far.

getComponentDescriptors

`public ComponentDescriptor[] getComponentDescriptors()`

Returns the set of descriptors of the connected components. Independent actions of the graph are grouped into a fictitious component whose id is `ComponentDescriptor.INDEPENDENT_COMPONENT`.

Returns:
the set of descriptors of the connected components.

fr.inria.gforge.telex.extensions
Class FastScheduleGenerator

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.extensions.ScheduleGenerator
      |
      +-fr.inria.gforge.telex.extensions.FastScheduleGenerator
```

public class **FastScheduleGenerator**
extends ScheduleGenerator

This ScheduleGenerator returns a schedule "on the fly", by exploring the underlaying RelationGraph. FIXME For the moment we cannot go further because the Schedule Class require arrays instead of list for actions, and non-actions. In the future, we plan to return a fake list that will iterate over the graph.
Author:
P.Sutra

Fields inherited from class `fr.inria.gforge.telex.extensions.ScheduleGenerator`
`_schedulingContext`, `DEFAULT_SCHEDULER`

Constructor Summary

`public`	`FastScheduleGenerator(SchedulingContext context)` This constructor is called by Telex.

Method Summary

`ScheduleGenerator`	`getInstance(DocumentImpl document)` Returns a schedule generator instance responsible for the specified document. Telex calls this method for each of the bound documents the graph to process relates to, even if it contains only one element. Telex passes the returned schedule generator to the execute() method.
`Schedule`	`getSchedule()`

Methods inherited from class `fr.inria.gforge.telex.extensions.ScheduleGenerator`
`getBaseStates, getBoundDocuments, getInstance, getParentScheduler, getStateGraph, isReleased, release, usesSplitGraph`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

FastScheduleGenerator

`public` **`FastScheduleGenerator`**`(SchedulingContext context)`

(continued from last page)

This constructor is called by Telex.

Parameters:

`context` - the scheduling context

Methods

getInstance

`public ScheduleGenerator **getInstance**(DocumentImpl document)`

Returns a schedule generator instance responsible for the specified document. Telex calls this method for each of the bound documents the graph to process relates to, even if it contains only one element. Telex passes the returned schedule generator to the execute() method.

getSchedule

`public Schedule **getSchedule**()`

fr.inria.gforge.telex.extensions
Class IterableScheduleGenerator

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.extensions.ScheduleGenerator
      |
      +-fr.inria.gforge.telex.extensions.IterableScheduleGenerator
```

All Implemented Interfaces:
java.lang.Iterable

---

public class **IterableScheduleGenerator**
extends ScheduleGenerator
implements java.lang.Iterable

A `ScheduleGenerator` that generates sound schedules upon request. Generating all possible schedules from a given action-constraint graph is CPU-consuming. Beside, the application may be interested in only a few or even just one schedule. These are the reasons for this schedule generator, whose purpose is to save CPU time by generating sound schedules dynamically, upon application request. The application may iterate through the generated schedules and stop when satisfied. This schedule generator complies with the guidelines listed in `ScheduleGenerator`.

Since an action-constraint graph may include actions of several (bound) documents, there are two types of (iterable) schedule generators: *primary* and *secondary*. Telex allocates a primary schedule generator for a given action-constraint graph. Its role is to actually compute global schedules from the graph. The primary generator then allocates a secondary generator for each bound document in the graph. Its role is to handle computation requests issued by the application editing the document and to forward them to the primary generator. When the secondary gets a global schedule from the primary, it filters out actions that do not belong to the document and passes the resulting schedule to the application. Note that to simplify coding, single-document action-constraint graphs are not treated as a special case: a primary and a secondary generator are allocated even in this case.

Iterable schedule generators may also be used in conjunction with `ComponentIterableScheduleGenerator`. In this case, the component-aware generator allocates one iterable generator for each connected component of the action-constraint graph and one iterator for the set of independent actions.

Author:
J-M. Busca INRIA/Regal

Fields inherited from class `fr.inria.gforge.telex.extensions.ScheduleGenerator`
`_schedulingContext`, `DEFAULT_SCHEDULER`

Constructor Summary

`public`	`IterableScheduleGenerator(SchedulingContext context)` Creates a primary iterable schedule generator on the specified scheduling context.
`public`	`IterableScheduleGenerator(SchedulingContext context, GraphComponent component)` Creates a primary iterable schedule generator on the specified scheduling context for the specified graph component.
`public`	`IterableScheduleGenerator(SchedulingContext context, java.util.List actions)` Creates a primary iterable schedule generator on the specified scheduling context for the specified set of independent actions.

public	`IterableScheduleGenerator(IterableScheduleGenerator primary, DocumentImpl document)` Creates a secondary iterable schedule generator for the specified document, linked to the specified primary generator.

Method Summary

`java.util.Set`	`getAbortedActions()` Returns the set of aborted actions.
`java.util.Set`	`getCommittedActions()`
`Schedule`	`getCommittedSchedule()` Returns the schedule that has been committed so far.
`ScheduleGenerator`	`getInstance(DocumentImpl document)`
`java.util.Iterator`	`iterator()` Returns an iterator over sound schedules.
`java.lang.String`	`toString()`

Methods inherited from class `fr.inria.gforge.telex.extensions.ScheduleGenerator`

`getBaseStates, getBoundDocuments, getInstance, getParentScheduler, getStateGraph, isReleased, release, usesSplitGraph`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface `java.lang.Iterable`

`iterator`

Constructors

IterableScheduleGenerator

```
public IterableScheduleGenerator(SchedulingContext context)
```

Creates a primary iterable schedule generator on the specified scheduling context. This constructor is called when the graph to process contains a single component.

Parameters:

`context` - the scheduling context to process.

IterableScheduleGenerator

```
public IterableScheduleGenerator(SchedulingContext context,
                                 GraphComponent component)
```

(continued from last page)

Creates a primary iterable schedule generator on the specified scheduling context for the specified graph component. This constructor is called when the graph to process is split into connected components. Each component is then handled by a specific iterable schedule generator.

Parameters:

`context` - the scheduling context to process.

`component` - the graph component to process.

IterableScheduleGenerator

```
public IterableScheduleGenerator(SchedulingContext context,
                                 java.util.List actions)
```

Creates a primary iterable schedule generator on the specified scheduling context for the specified set of independent actions. This constructor is called when the graph to process is split into connected components. The set of independent actions is processed as a fictitious component.

Parameters:

`context` - the scheduling context to process.

`actions` - the set of independent actions to process.

IterableScheduleGenerator

```
public IterableScheduleGenerator(IterableScheduleGenerator primary,
                                 DocumentImpl document)
```

Creates a secondary iterable schedule generator for the specified document, linked to the specified primary generator.

Parameters:

`primary` - the primary generator to link the secondary generator to.

`document` - the document the secondary generator handles.

Methods

toString

```
public java.lang.String toString()
```

---

getInstance

```
public ScheduleGenerator getInstance(DocumentImpl document)
```

Returns a schedule generator instance responsible for the specified document. Telex calls this method for each of the bound documents the graph to process relates to, even if it contains only one element. Telex passes the returned schedule generator to the execute() method.

---

getCommittedSchedule

```
public Schedule getCommittedSchedule()
```

Returns the schedule that has been committed so far.

Returns:

the schedule that has been committed so far.

(continued from last page)

getCommittedActions

`public java.util.Set` **`getCommittedActions`**`()`

getAbortedActions

`public java.util.Set` **`getAbortedActions`**`()`

Returns the set of aborted actions.

Returns:
the set of aborted actions.

iterator

`public java.util.Iterator` **`iterator`**`()`

Returns an iterator over sound schedules. Calling the `Iterator.hasNext()` method on the returned iterator actually compute a new sound schedule, if any. Calling the `Iterator.next()` method retrieves the schedule just computed. The `Iterator.remove()` method is not supported: a call to this method throws the UnsupporteOperationException.

Returns:
an iterator over sound schedules.

fr.inria.gforge.telex.extensions
Class ScheduleGenerator

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.extensions.ScheduleGenerator
```

Direct Known Subclasses:
FastScheduleGenerator, IterableScheduleGenerator, ComponentIterableScheduleGenerator

---

public abstract class **ScheduleGenerator**
extends java.lang.Object

An object that generates sound `Schedule`s. Telex passes such an object to a `TelexApplication` as parameter to the `TelexApplication.execute(Document, ScheduleGenerator)` method when the specified document is updated, as described in `SchedulingParameters`. The application can then learn about the new state of the document by invoking the generator. When the application is finished, it must release the generator to indicate that it is ready to accept new generators.

Sound schedule set. In the general case, several sound schedules exist for a given set of actions and constraints. Thus a schedule generator should not generate just one schedule, but rather a set of alternative schedules. Indeed, the application must present these alternatives to user so that he can choose the one he prefers. A schedule generator should comply as much as possible with the following guidelines, by order of priority:

- only one of each set of equivalent schedules (according to *non-commuting* constraints) should be handed to the application,
- in case of conflict between actions of local user and that of remote users, schedules containing actions of local user should be handed to the application first,
- if the application has previously specified one or more *preferred* schedules, newly-generated schedules should be prefixed with one of the preferred schedules whenever possible,
- schedules should include as many of the actions of the action-constraint graph as possible.

Bound documents. A schedule generator is passed the action-constraint graph which it must compute sound schedules from. Most often, the actions of this graph belong to only one document. A schedule generator, however, must handle the case in which the graph contains the actions of several bound documents. See below the use of the `getInstance(DocumentImpl)` method.

Connected components. An action-constraint graph, whether related to a single document or to a set of bound documents, may have several connected components. By definition, the schedules generated on distinct components are independent. For a better experience, the user should be presented a set of independent alternatives, so that he can specify his choice for each of them. Schedule generators offering this capability may request Telex to provide them with a split `fr.inria.gforge.telex.scheduling.RelationGraph` by overriding the `usesSplitGraph()` method.

Implementation and extensions.Telex provides a default general-purpose generator: `DEFAULT_SCHEDULER`. A `TelexApplication` with specific needs may replace this default generator with its own by specifying it in its `ProcessingParameters`. The application-specific generator must extend the `ScheduleGenerator` class and provide a public constructor with the (SchedulingContext) signature.

Whenever new sound schedules must be computed, Telex allocates a new schedule generator by calling its constructor and passing it the current scheduling context. It then calls the `getInstance(DocumentImpl)` method to get a document-specific generator for each of the bound documents the graph relates to. Finally, Telex calls the `TelexApplication.execute(Document, ScheduleGenerator)` method to pass each generator to the corresponding application. Note that (i) Telex calls the getInstance() method even if the action-constraint graph contains actions from only one document, (ii) when opening a document, Telex first calls the constructor of the schedule generator with a null scheduling context and then calls the usesSplitGraph() method to determine whether it should maintain a split graph or not.

Author:
J-M. Busca INRIA/Regal

Field Summary

`protected`	`_schedulingContext` The execution context of the schedule generator.
`public static final`	`DEFAULT_SCHEDULER` The default schedule generator of Telex.

Constructor Summary

`protected`	`ScheduleGenerator()` Creates a schedule generator.
`protected`	`ScheduleGenerator(SchedulingContext context)` FIXME PIERRE I don't think that with the current architecture we still need the SchedulingContext class; we should pass directly the scheduler.

Method Summary

`DocumentState[]`	`getBaseStates()` Convenience method equivalent to _schedulingContext.getBaseStates().
`DocumentImpl[]`	`getBoundDocuments()` Convenience method equivalent to _schedulingContext.getBoundDocuments().
`abstract ScheduleGenerator`	`getInstance(DocumentImpl document)` Returns a schedule generator instance responsible for the specified document.
`Scheduler`	`getParentScheduler()` Convenience method equivalent to _schedulingContext.getParentScheduler().
`RelationGraph`	`getStateGraph()` Convenience method equivalent to _schedulingContext.getStateGraph().
`boolean`	`isReleased()` Returns whether this schedule generator has been released.
`void`	`release()` Releases this schedule generator.
`boolean`	`usesSplitGraph()` Returns whether this generator uses a split graph.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

DEFAULT_SCHEDULER

`public static final java.lang.Class` **`DEFAULT_SCHEDULER`**

The default schedule generator of Telex. This constant is the `IterableScheduleGenerator` class.

(continued from last page)

_schedulingContext

`protected fr.inria.gforge.telex.extensions.SchedulingContext` **`_schedulingContext`**

The execution context of the schedule generator.

Constructors

ScheduleGenerator

`protected` **`ScheduleGenerator`**`()`

Creates a schedule generator.

ScheduleGenerator

`protected` **`ScheduleGenerator`**`(SchedulingContext context)`

FIXME PIERRE I don't think that with the current architecture we still need the SchedulingContext class; we should pass directly the scheduler. Creates a schedule generator on the specified scheduling context. Note that the context may be null. In that case, Telex will only calls the `usesSplitGraph()` method on the created object.

Parameters:

`context` - the scheduling context to process.

Methods

getInstance

`public abstract ScheduleGenerator` **`getInstance`**`(DocumentImpl document)`

Returns a schedule generator instance responsible for the specified document. Telex calls this method for each of the bound documents the graph to process relates to, even if it contains only one element. Telex passes the returned schedule generator to the execute() method.

Parameters:

`document` - the document for which to return a schedule generator instance.

Returns:

a schedule generator instance.

---

usesSplitGraph

`public boolean` **`usesSplitGraph`**`()`

Returns whether this generator uses a split graph. The `fr.inria.gforge.telex.scheduling.Scheduler` calls this method to allocate an action-constraint graph of the appropriate type. The default implementation of this method returns false.

Returns:

true if this schedule generator uses a split graph and false otherwise.

---

release

`public final void` **`release`**`()`

Releases this schedule generator. A `TelexApplication` must call this method in order to receive new schedule generators for the same document.

isReleased

`public final boolean **isReleased**()`

Returns whether this schedule generator has been released.

Returns:
true if this schedule has been released, and false otherwise.

getParentScheduler

`public Scheduler **getParentScheduler**()`

Convenience method equivalent to _schedulingContext.getParentScheduler().

Returns:
the value of _schedulingContext.getParentScheduler().

getStateGraph

`public RelationGraph **getStateGraph**()`

Convenience method equivalent to _schedulingContext.getStateGraph().

Returns:
the value of _schedulingContext.getStateGraph().

getBoundDocuments

`public DocumentImpl[] **getBoundDocuments**()`

Convenience method equivalent to _schedulingContext.getBoundDocuments().

Returns:
the value of _schedulingContext.getBoundDocuments().

getBaseStates

`public DocumentState[] **getBaseStates**()`

Convenience method equivalent to _schedulingContext.getBaseStates().

Returns:
the value of _schedulingContext.getBaseStates().

fr.inria.gforge.telex.extensions
Class SchedulingContext

```
java.lang.Object
  |
  +-fr.inria.gforge.telex.extensions.SchedulingContext
```

public final class **SchedulingContext**
extends java.lang.Object

The execution context of a `ScheduleGenerator`. It contains (i) the `fr.inria.gforge.telex.scheduling.Scheduler` from which the context is drawn and (ii) the `ScheduleGenerator` that this scheduler previously generated. It also contains a snapshot of the following items:

- the action-constraint graph,
- the document(s) this graph corresponds to,
- the base state of each of these documents,
- the action filters of each of these documents.

Author:
J-M. Busca INRIA/Regal

Constructor Summary

`public`	`SchedulingContext(Scheduler scheduler, boolean isNew)` Creates a new execution context drawn from the specified scheduler.

Method Summary

`DocumentState[]`	`getBaseStates()` Returns the snapshot of the base states of the bound documents.
`DocumentImpl[]`	`getBoundDocuments()` Returns the snasphot of bound documents related to the state graph.
`DocumentImpl[]`	`getMainDocuments()` Returns the snapshot of main documents related to the state graph.
`Scheduler`	`getParentScheduler()` Returns the scheduler from which this context is drawn.
`ScheduleGenerator`	`getPreviousGenerator()` Returns the previous schedule generator allocated by the parent scheduler.
`RelationGraph`	`getStateGraph()` Returns the snapshot of the state (action-constraint) graph to process.
`java.lang.String`	`toString()`

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

(continued from last page)

Constructors

SchedulingContext

```
public SchedulingContext(Scheduler scheduler,
                         boolean isNew)
```

Creates a new execution context drawn from the specified scheduler. This constructor records the scheduler and the previous schedule generator. It then takes a snapshot of all of the items listed above. It then applies to the snapshot of the graph all active action filters.

This constructor must be called within a synchronized section so that all items contained in the context are both consistent and mutually consistent.

Parameters:

`scheduler` - the scheduler from which the context is drawn.

`isNew` - true if data structures must be cloned.

Methods

toString

```
public java.lang.String toString()
```

---

getParentScheduler

```
public Scheduler getParentScheduler()
```

Returns the scheduler from which this context is drawn.

Returns:

the scheduler from which this context is drawn.

---

getPreviousGenerator

```
public ScheduleGenerator getPreviousGenerator()
```

Returns the previous schedule generator allocated by the parent scheduler.

Returns:

the previous schedule generator allocated by the parent scheduler.

---

getStateGraph

```
public RelationGraph getStateGraph()
```

Returns the snasphot of the state (action-constraint) graph to process.

Returns:

the snasphot of the state (action-constraint) graph to process.

---

getBoundDocuments

```
public DocumentImpl[] getBoundDocuments()
```

Returns the snasphot of bound documents related to the state graph.

(continued from last page)

Returns:
the snasphot of bound documents related to the state graph.

getMainDocuments

```
public DocumentImpl[] getMainDocuments()
```

Returns the snasphot of main documents related to the state graph.

Returns:
the snasphot of main documents related to the state graph.

getBaseStates

```
public DocumentState[] getBaseStates()
```

Returns the snapshot of the base states of the bound documents.

Returns:
the snapshot of the base states of the bound documents.

Index