Agent Interaction SDK Services Developer Guide

The Agent Service

4/25/2026

Contents

- 1 The Agent Service
  - 1.1 Introduction
  - 1.2 Agent Service Essentials
  - 1.3 Forms and Agent Actions

The Agent Service

The agent service is the `IAgentService` interface defined in the `com.genesyslab.ail.ws.agent` namespace. To be able to manage agent features, your application has to integrate this interface and uses classes.

Introduction

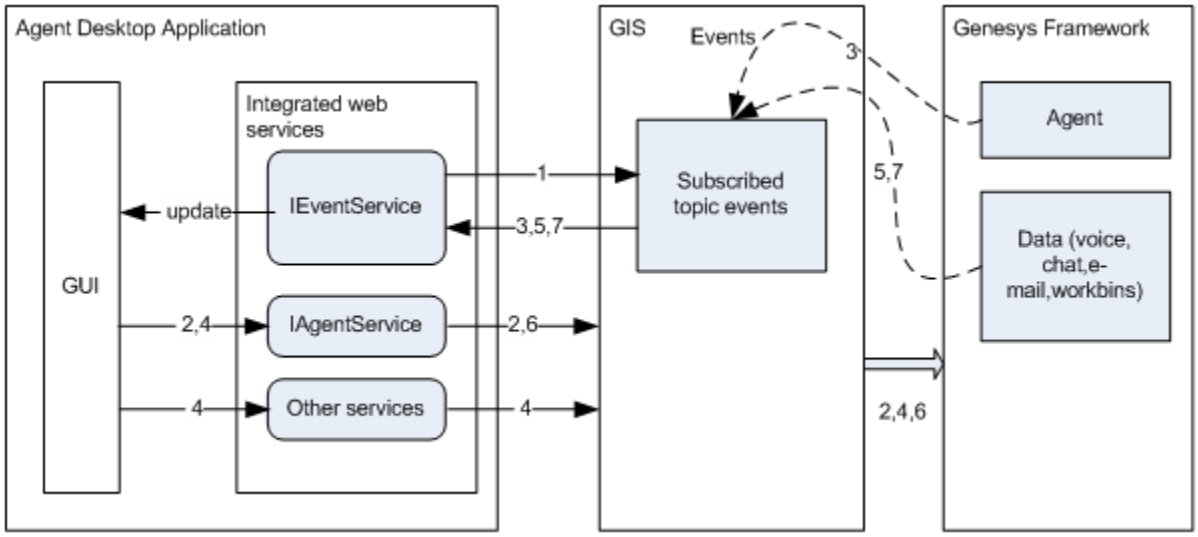
The agent service is used to access the agent features managed on the server-side application. This service is required for most of the applications developed with the Agent Interaction Service API. Without it, you cannot access most of the features with others services that require an identified agent.

For example, if your application has to make a call, it requires an `IInteractionVoiceService` to access to a DN performing the call (see Voice Interactions). To access to this DN, your application needs to login an agent on the DN. Therefore your application needs to integrate an agent service.

To use correctly the agent service, your application must take into account integrated services events including agent services events when subscribing `TopicsEvent` with the `IEventService` interface. (See The Event Service).

Then, your application needs to log in an agent with the `IAgentService` interface. If the login is successful, your application receives the corresponding events and is able to use other services features, including agent service features. Services actions—corresponding to the Agent Interaction Service features—generate events that are taken into account according to the current `TopicsEvent` subscribed by your `IEventService`. Once your agent service logs out agents, services that depend on a logged in agent do not fulfill action requests.

The figure below demonstrates a typical request and event flow.



The Integrated Agent Service

Subscribing to TopicsEvent for integrated services, including agent service.

1. Requesting an agent login with the agent service login action.
2. Receiving an agent event for a successful agent login.
3. Requesting services actions depending on the logged agent, including agent services actions (ready, not ready, and so on).
4. Receiving events due to services actions.
5. Requesting an agent logout with the agent service logout action.
6. Receiving an agent event for logout.

The Integrated Agent Service shows the general agent service handling in an application integrating services depending on logged agents. As you can see, the agent service plays a predominant and determinant role for services requiring an agent.
The agent service is designed to:

- Access all the data related to a set of agents.
- Access agents status and possible agents actions for any logged agent.
- Perform agent actions on media, such as login, logout, ready, not ready, or aftercallwork actions.

A characteristic of this agent service is that voice media have dedicated classes to handle voice media data, whereas other media–chat and e-mail–use a set of common media classes.

Moreover, some agent actions require a filled form handling action data.

Agent Service Essentials

This section introduces essential aspects for using features of the agent service needed in most agent desktop applications developed in the Agent Interaction Service API.

Agent and Statuses

The agent status is not a general status of an agent. Rather, an agent status is related to a medium, that is an agent status is the status of an identified agent for a particular medium.

Using an Agent Status

The agent media status is modified:

- Each time an action requested by the agent service is performed on the media.
- When the media is impacted by a performed action that another service requested.
- When the media is affected by an internal change (for example, a system issue).

In terms of development requirements, take into account agent media statuses so as to:

- Inform your agent of its current status on a media and of any status change.
- Base on statuses your application design.

Agent Status for Voice Media

Voice media involve underlying switches and imply a dedicated management taken into account by the voice state model.

The Existing Voice Media Statuses

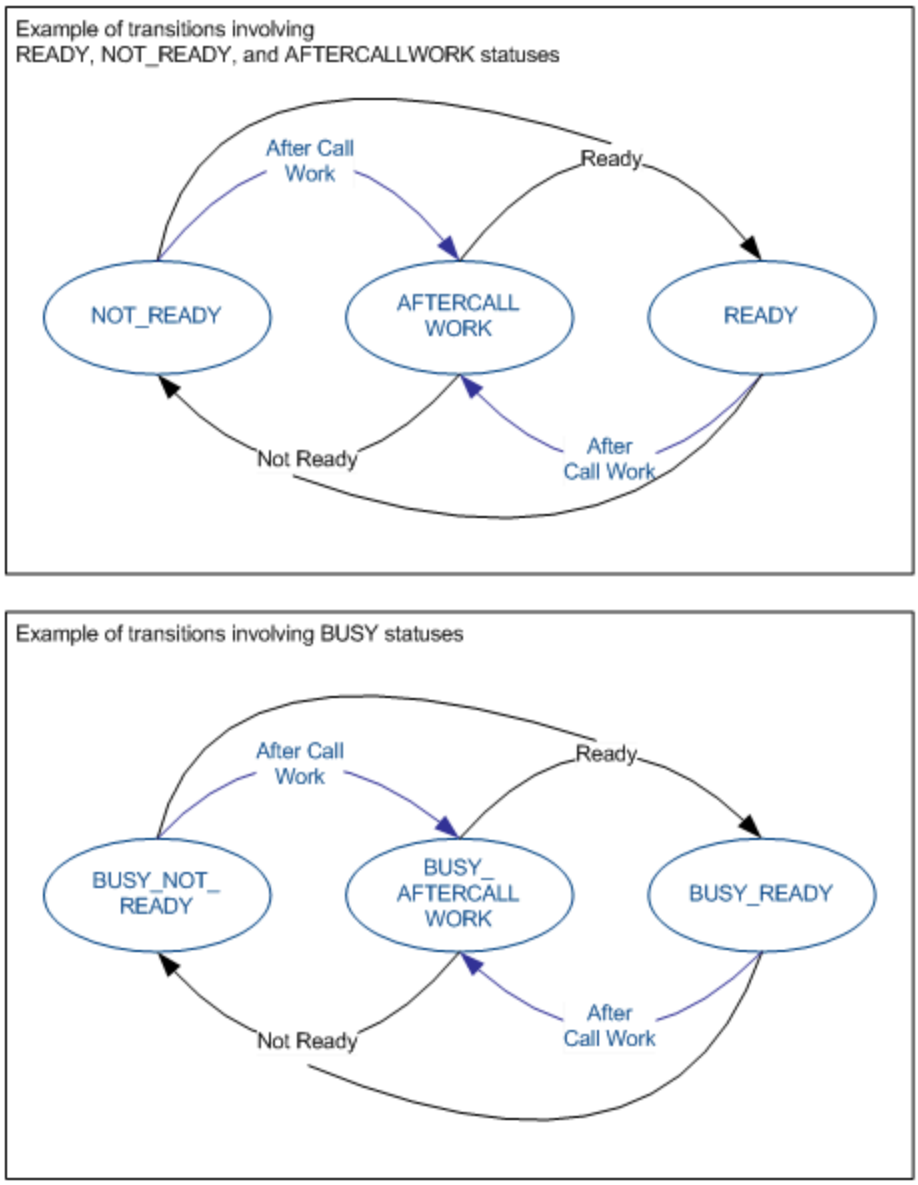
The `VoiceMediaStatus` type is an enumerated type describing the agent status on a DN:

- `LOGGED_OUT`—the agent is logged out.
- `READY`—The agent is logged and ready to work on the DN.
- `BUSY_READY`—The agent is busy.
- `NOT_READY`—The agent is logged and not ready for working.
- `BUSY_NOT_READY`—The agent is busy and not ready.
- `BUSY_LOGGED_OUT`—The agent is busy and logged out.
- `AFTERCALLWORK`—The agent is in After Call Work mode.
- `BUSY_AFTERCALLWORK`—The agent is busy and in After Call Work mode.
- `OUT_OF_SERVICE`—The DN is out of service.

> **Important**
>
> The `VoiceMediaStatus` type is defined in the `com.genesyslab.ail.ws.place` namespace as it is a media information. See Place, DNs, and Media for further details.

> **Warning**
>
> For voice media, some statuses depend on the switch capability. For example, the `AFTERCALLWORK` status is usable only if the switch offers this feature.

The following figure shows examples of the state transition that can occur due to calls to agent service features.



Examples of State Diagrams for Voice Media

> ## Warning
> This figure is provided as an informative example. It does not include all the existing transitions.

The transition from one state to another is not always due to a performed agent service action. State changes can occur due to other reasons, as for example, a T-Server event, or internal management

on the server-side application.

> **Warning**
> Do not assume any status sequence in your application design. Design your application always to update with the provided possible agent actions and current agent status.

Dealing with Voice Media Statuses

The statuses are handled in the `VoiceMediaInfo` class of the`com.genesyslab.ail.ws.place` namespace. This class associates the status `VoiceMediaInfo`.status with a DN identifier `VoiceMediaInfo`.dnId.
You can retrieve `VoiceMediaInfo` objects with the following attributes of the `IAgentService` interface:

- `agent:dns`—`VoiceMediaInfo` array containing all the DNs of the current place of the agent.
- `agent:loggedDns`— `VoiceMediaInfo` array containing all the DNs where the agent is logged.

Getting these attributes values is done with agent DTO classes of the `com.genesyslab.ail.ws.agent` namespace as shown in the following code snippet:

```
/// Retrieving the DTOs
PersonDTO[] agentDTO = myAgentService.getPersonsDTO(new string[]{ "agent0" }, new string[]{
"agent:loggedDns"});
///Getting the VoiceMediaInfo array
VoiceMediaInfo[] arrayInfo = (VoiceMediaInfo[]) agentDTO[0].data[0].value ;

/// Displaying the status for each DN
foreach(VoiceMediaInfo info in arrayInfo)
{
   System.Console.WriteLine("agent0 status on "+ info.dnId +" is
"+info.status.ToString()+"\n");
}
```

> **Important**
> Attributes related to statuses are published in events. You should update your agent desktop application with the statuses of the agent when a `VoiceMediaEvent` occurs with a matching `STATUS_CHANGED` filter.

Agent Status for Other Media

Chat and e-mail are standard media and their agent media statuses management is similar to agent voice media statuses.

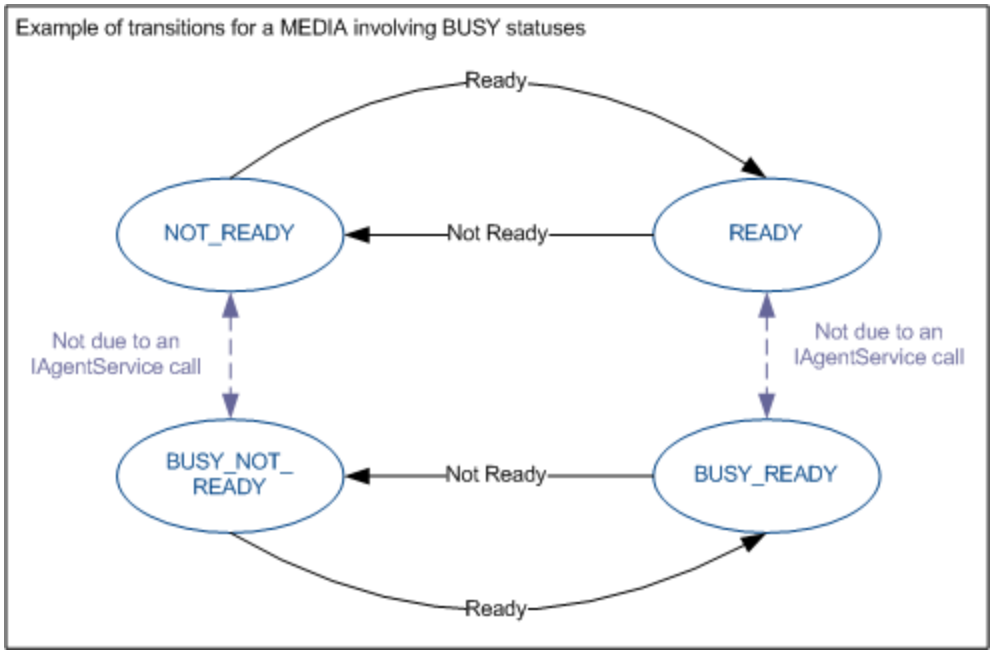
The Existing Media Statuses

The`MediaStatus` type is an enumerated type describing the agent status on a medium:

- `LOGGED_OUT`—The agent is logged out.
- `READY`—The agent is logged and ready to work with the media.
- `BUSY_READY`—The agent is busy.
- `NOT_READY`—The agent is logged and not ready for working.
- `BUSY_NOT_READY`—The agent is busy and not ready.
- `OUT_OF_SERVICE`—The medium is out of service.

> **Important**
>
> The MediaStatus type is defined in the com.genesyslab.ail.ws.place namespace as it is a media information. See Place, DNs, and Media.

The diagram example below shows examples of state transitions that can occur due to an `IAgentService` feature call.



An Example of Agent Media Status Diagram

> ## Warning
> This figure is provided as an informative example. It does not include all the existing statuses and transitions.

The transition from one state to another is not always due to a performed agent service action. State changes can occur due to other reasons, as for example, a T-Server event, or internal management on the server-side application.

> ## Warning
> Do not assume any status sequence in your application design. Design your application always to update with the provided possible agent actions and current agent status.

Dealing with Media Statuses

The `MediaInfo` class of the`com.genesyslab.ail.ws.place` namespace associates the status `MediaInfo.status` with a media name `MediaInfo.name`.

> **Important**
> Possible values for the MediaInfo.name attribute are chat or email.

You can retrieve MediaInfo objects with the following attributes of the IAgentServiceinterface:

- agent:availableMedias—array of the mediaType available on the current place of the agent.
- agent:loggedMedias—MediaInfo array containing all the media on which the agent is logged.

Dealing with MediaInfo objects is performed with agent DTO classes of the com.genesyslab.ail.ws.agent namespace as shown in the following code snippet:

```
/// Retrieving the DTOs
PersonDTO[] agentDTO = myAgentService.getPersonsDTO(new string[]{ “agent0” }, new string[]{
"agent:loggedMedias"});
///Getting the MediaInfo array
MediaInfo[] arrayInfo = (MediaInfo[]) agentDTO[0].data[0].value ;

/// Displaying the status for each media
foreach(MediaInfo info in arrayInfo)
{
  System.Console.WriteLine("agent0 status on "+ info.name +" is "+
info.status.ToString()+"\n");
}
```

> **Important**
> The IAgentService attributes are published in events. You should update your agent desktop application with the statuses of the agent when a MediaEvent occurs with a matching STATUS_CHANGED filter.

Agent and Possible Actions

In the agent service, voice media actions are differentiated from other media actions. There are two enumerated types defining the existing actions for an agent:

- AgentDnActions —actions that the IAgentService interface can perform on a voice medium, that is, a DN.
- AgentMediaActions—actions that the IAgentService interface can perform on media such as chat and e-mail.

Each action defined in these enumerated types is associated with a method of the IAgentService interface. For example, AgentDnAction.LOGIN represents the IAgentService.login() method that is able to perform the login of an agent on a set of media.
The following sub-sections details possible agent action use with the IAgentService interface.

Understanding Agent Possible Action

The `IAgentService` interface lets your application retrieve the possible actions associated with a media. You can use this information to enable or disable your desktop application's features that use the corresponding calls to your service.
For example, if at a certain point in time a ready action is not possible for a given medium, your application should disable its ready button if the user selects this medium.
The possible actions take into account:

- The current state of the media concerned by the action.
- The capability of the underlying genesys solution to realize the action.

You retrieve these possible actions with DTOs as explained in the following subsections.

> **Warning**
> It is not possible to retrieve the attribute values corresponding to possible actions on media on which no agent is logged.

Voice Media Possible Actions

The possible actions for a voice media are accessed with the `agent:dnsActionsPossible` attribute of the `IAgentService` interface. This attribute contains an array of `DnActionsPossible` associating the current available actions with DN identifiers.
The following code snippet shows how to display the voice available actions for `agent0` using DTO classes of `com.genesyslab.ail.ws.agent` namespace:

```
/// Retrieving the DTOs
PersonDTO[] agentDTO = myAgentService.getPersonsDTO(new string[]{ "agent0" }, new
string[]{"agent:dnsActionsPossible"});

DnActionsPossible[] myPossibleActionsOnDN = (DnActionsPossible[]) agentDTO[0].data[0].value ;

/// Displaying the Possibilities for each DN.
foreach(DnActionsPossible myPossibleActions in myPossibleActionsOnDN)
{
   System.Console.WriteLine("agent0 possible actions on " + myPossibleActions.dnId +" are:");
   foreach(AgentDnAction action in myPossibleActions.agentActions)
   {
      System.Console.Write("\t"+action.ToString());
   }
   System.Console.WriteLine("\n");
}
```

Other Media Possible Actions

The possible actions for other media are accessed with the `agent:mediaActionsPossible` attribute of the `IAgentService` interface. This attribute contains an array of `MediaActionsPossible` associating the current available actions with media names.

> **Important**
> The authorized values for media names are `chat` and `email`.

The following code snippet shows how to display the available actions for `agent0` using DTO classes of `com.genesyslab.ail.ws.agent` namespace:

```
/// Retrieving the DTOs
PersonDTO[] agentDTO = myAgentService.getPersonsDTO(new string[]{ "agent0" }, new
string[]{"agent:mediaActionsPossible"});

MediaActionsPossible[] myPossibleActionsOnMedia = (MediaActionsPossible[])
agentDTO[0].data[0].value ;

/// Displaying the possible actions for each media.
foreach(MediaActionsPossible myPossibleActions in myPossibleActionsOnMedia)
{
   System.Console.WriteLine("agent0 possible actions on " + myPossibleActions.mediaType +"
are:");
   foreach(AgentMediaAction action in myPossibleActions.agentActions)
   {
      System.Console.Write("\t"+action.ToString());
   }
   System.Console.WriteLine("\n");
}
```

Agent and Events

There are two types of agent event related to media:

- `VoiceMediaEvent`—agent events occurring on voice media.
- `MediaEvent`—agent events occurring on chat or e-mail media.

These events enable application update based on agent, place, voice, and media events. The available attributes through these events let your application:

- Update the agent media statuses with:
  - `agent:mediaInfo` for a media
  - `agent:voiceMediaInfo` for a DN.
- Update the possible agent actions for a medium with:
  - `agent:dnActionsPossible` for a single DN
  - `agent:mediaActionsPossible` for another medium such as e-mail.

  The `PlaceChangedEvent` occurs when an agent changes place. This lets your application update GUI components displaying information related to the place.

> **Important**
>
> The place is an object associating an agent with several media. See Place, DNs, and Media

Forms and Agent Actions

The Agent Interaction Service API includes form classes to handle the data related to a medium. Voice is managed differently from other media, such as chat and e-mail.
For most agent actions on media, your application has to define a form containing the parameters required for perform the actions.
The following subsections detail the existing forms and show their use in agent service features.

Forms for Voice Media

The forms used for voice media are classes of `com.genesyslab.ail.ws.agent` namespace. There are several form objects that may be defined in a voice context:

- `LoginVoiceForm`—This class is used to login on voice DNs.
- `LogoutVoiceForm`—This class is used to logout on voice DNs.
- `ReadyVoiceForm`—This class is used to get ready or not ready on voice DNs.
- `AfterCallWorkVoiceForm`—This class is used when getting in an AfterCallWork status on voice DNs.

Each form is dedicated to an action. However, the data are characteristics of the voice media. For example, the LoginVoiceForm class is used to fill forms required for a login and includes the whole list of the data involved in voice forms. The following code snippet shows how to fill this form.

```
// Creating the voice form for the login of the agent0
LoginVoiceForm myLoginVoiceForm = new LoginVoiceForm();

// Defining the set of DNs concerned with the login
myLoginVoiceForm.dnIds = new string[]{ "DN0" };

// Setting the login ID of the agent0
myLoginVoiceForm.loginId = "login0";

// Setting the agent password
myLoginVoiceForm.password = "Password0";

// Setting the queue parameter
myLoginVoiceForm.queue = "myQueue";

// Optional: you can set reasons otherwise null
myLoginVoiceForm.reasons = null;

// You can tune your application with switch specific data
myLoginVoiceForm.TExtensions = null;
```

```
// You can specify the workmode for your login.
myLoginVoiceForm.workmode = WorkmodeType.MANUAL_IN;
```

For further information on each form attributes, see the *Agent Interaction SDK 7.6 Services API Reference.*

T-Extensions

T-Extensions are data used by the underlying switches. T-Extensions are switch specific, you can fine-tune your application with these data for a specific switch type.
Use a KeyValue array to specify T-Extension keys-values as shown in the following code snippet:

```
// Specifying a trunk for the G3 Lucent switch
KeyValue[] myTExtensions = new KeyValue[1];
myTExtensions[0]=new KeyValue();
myTExtensions[0].key = “Trunk”;
myTExtensions[0].value = “myTrunk”;
```

> **Important**
>
> For further information about key-value T-Extensions, refer to the T-Server documentation associated with your switch. See also T-Extensions and T-Reasons.

Workmode

Workmode is switch specific. Switches don’t always provide all the workmodes defined by the `com.genesyslab.ail.ws.agent.WorkmodeType` enumeration. Refer to your T-Server documentation for further information.

> **Important**
>
> The workmode can affect the state sequence of the agent. See also Workmode.

Forms for Other Media

Other media -e-mail or chat- require the `MediaForm` class of the `com.genesyslab.ail.ws.agent` namespace to perform agent service actions, such as login, logout, ready, and so on.
This form is a container which specify the media types concerned by the agent service actions.
The following code snippet shows how to define this form for chat and e-mail.

```
// Creating an instance of MediaForm
MediaForm myMediaForm = new MediaForm();
// Defining the types of concerned media
myMediaForm.mediaTypes = new string[]{ "email", "chat" };
```

Agent Login

The `IAgentService` interface has a `login()` method to let agents login on a place for a set of

media defined in their corresponding forms. See Forms for Voice Media and Forms for Other Media.
The following code snippet shows a login of the agent0 agent performed on an authorized place, using the myLoginVoiceForm and myMediaForm that were defined in the previous sections:

```
MediaInfoError[] errors = myAgentService.login( "agent0", "placeForAgent0",
myLoginVoiceForm, myMediaForm);
```

The login is performed on the set of DNs defined in the LoginVoiceForm and on the media that are defined in the place and specified in the types of the MediaForm.
You retrieve in MediaInfoErrors objects both errors and useful information about the login attempt.

Agent Logout

The IAgentService interface has a logout() method to let agents logout a place. The agent performs a logout only for the set of media defined in their corresponding forms.
The following code snippet is an example of a logout performed by the agent0 agent.

```
// Creating a instance of LogoutVoiceForm
LogoutVoiceForm myLogoutVoiceForm = new LogoutVoiceForm();

// Defining the set of IDs concerned by the logout
myLogoutVoiceForm.dnIds = new string[]{ "DN0" };

// Setting the queue and the reason attributes
myLogoutVoiceForm.queue = “myQueue”;
myLogoutVoiceForm.reasons = null;

// Logout action of the agent0 performed on DN0 // and the media defined in myMediaForm
MediaInfoError[] errors= myAgentService.logout( "agent0", myLogoutVoiceForm,
myMediaForm);
```

Getting Ready or Not Ready

The IAgentService interface has ready() and notReady() methods to let agents get ready or not ready on media defined in a place. The agent performs the ready and not ready actions on the set of media defined in their corresponding forms.
The following code snippet is an example of a ready action performed by the agent0 agent.

```
// Creating a instance of ReadyVoiceForm
ReadyVoiceForm myReadyVoiceForm = new ReadyVoiceForm();
// Defining the set of IDs concerned by the ready (or not ready)
myReadyVoiceForm.dnIds = new string[]{ "DN0" };

// Setting the other attributes
myReadyVoiceForm.queue = "myQueue";
myReadyVoiceForm.reasons = null;
myReadyVoiceForm.TExtensions=null;
myReadyVoiceForm.workmode = WorkmodeType.MANUAL_IN;

// Agent0 getting ready on DN0 // and the media defined in myMediaForm
MediaInfoError[] errors =
myAgentService.ready("agent0", myReadyVoiceForm, myMediaForm);
```

The notReady() call is equivalent to the ready() call as illustrated in the following code snippet:

```
MediaInfoError[] errors = myAgentService.notReady("agent0", myReadyVoiceForm,
 myMediaForm);
```

Managing Agent Skills

In 7.6.6, the Agent Service allows to manage Agent skills (add, update, or remove) using the following method:

```
void updateAgentSkills(
   string agentId,
   SkillDTO[] addedSkills,
   SkillDTO[] updatedSkills,
   int[] deletedSkills);
```

> **Important**
>
> To manage skills, the skills must exist in the Configuration Server.

The following code sample shows how you should proceed:

```
//get all skills available in the configuration server
 Skill[] skills = resourceService.getAvailableSkills();
 //Create a Skill DTO
 SkillDTO dto = new SkillDTO();
 // Set the skill level
 dto.level = 11;
 // Get the skill's name in the Configuration Server
 dto.name = skills[0].name;
 // Get the skill's config DB ID
 dto.skillId = Convert.ToInt32(skills[0].id);
 //Now assign a new skill to the given agent
 agentService.updateAgentSkills(anAgent1, new SkillDTO[] { dto }, null, null);
```