Oracle Service Cloud  
Multi-Channel Toolkit

Developer Startup Guide

Table of Contents

[Introduction 3](#_Toc341099181)

[Audience 4](#_Toc341099182)

[Assemblies 5](#_Toc341099183)

[Oracle.RightNow.Cti 5](#_Toc341099184)

[Oracle.RightNow.Cti.MediaBar 5](#_Toc341099185)

[Oracle.RightNow.Cti.ScreenPopHandlers.Samples 5](#_Toc341099186)

[Oracle.RightNow.Cti.Providers.Simulator 5](#_Toc341099187)

[Core Types 6](#_Toc341099188)

[Classes 6](#_Toc341099189)

[Interfaces 6](#_Toc341099190)

[Enums 6](#_Toc341099191)

[The Extensibility Model 7](#_Toc341099192)

[Imported Types 7](#_Toc341099193)

[Exported Types 7](#_Toc341099194)

[Implementing a Provider 9](#_Toc341099195)

[Creating the Interaction Provider 9](#_Toc341099196)

[The Interaction Types 10](#_Toc341099197)

[The Agent 10](#_Toc341099198)

[The Device 10](#_Toc341099199)

[Summary 10](#_Toc341099200)

[The Sample Provider 12](#_Toc341099201)

[Screen Pop 13](#_Toc341099202)

[Known Issues 14](#_Toc341099203)

Introduction

The Oracle Multi-Channel Toolkit (referred to as *toolkit provides* a framework to create CTI (Computer Telephony Integration) solutions for the Oracle Service Cloud platform. The toolkit facilitates the implementation of common patterns and enables a common CTI experience for agents and administrators.

Some of the features provided by the toolkit are:

* An extensible and customizable media bar
* Automatic active record synchronization with contact number detection for outbound dialing
* Support for multiple simultaneous interactions
* Support for voice, email, web and generic incidents
* Built in transfer support
* Interaction and agent state control
* Extensible screen pop capabilities, with built in screen pop handlers

The goal of this document is to help developers get started building providers for the Oracle Service Cloud Multi-Toolkit. In this document, we’ll outline the extensibility model and walk through examples of how to implement the various components used by the toolkit.

Audience

This document targets software developers, software architects, CTI and telephony professionals.

This documentation assumes that the reader is familiar with .NET and the RightNow Connect Add-In Framework. A good understanding of CTI is also helpful and deep coverage of those topics are outside of the scope of this guide.

Assemblies

The toolkit is composed of several assemblies. A good understanding of the role played by each assembly is helpful in understanding how the toolkit may be extended and how out of the box functionality may be replaced.

Oracle.RightNow.Cti

This is the core toolkit assembly and it defines the object model types, common component interfaces, configuration and helper types.

Oracle.RightNow.Cti.MediaBar

This the assembly where the default media bar is implemented. If you intend to replace the UI used by the toolkit, you would replace this assembly with your own implementation.

Oracle.RightNow.Cti.ScreenPopHandlers.Samples

This the assembly contains sample implementations of screen pop handlers.

Oracle.RightNow.Cti.Providers.Simulator

This is where the sample interaction provider is implemented. The example connects to a simulated environment and allow interaction simulation. The types implemented in this assembly will be the primary focus of this guide, as they are the core components driving the behavior of the toolkit.

Core Types

Before we dive into the interaction provider implementation, let’s take a look at the types that define the object model and core interfaces exposed by the toolkit. All of the types in the following table are defined in the Oracle.RightNow.Cti assembly.

Classes

|  |  |
| --- | --- |
| Type | Description |
| Oracle.RightNow.Cti.Model.AgentState | Represents an agent state. |
| Oracle.RightNow.Cti.Model.Contact | Represents a contact. This usually represents a target for a dial, transfer or conference operation. |
| Oracle.RightNow.Cti.Model.CtiNotification |  |
| Oracle.RightNow.Cti.Model.InteractionCredentials |  |
| Oracle.RightNow.Cti.Model.LocationInfo | Information about the location. This is a combination of station (desktop) and agent extension or device. |
| Oracle.RightNow.Cti.Model.SwitchCredentials | Represents an agent’s credentials on the switch. |

Interfaces

|  |  |
| --- | --- |
| Type | Description |
| Oracle.RightNow.Cti.Model.ICall | Represents a call interaction. |
| Oracle.RightNow.Cti.Model.IDevice | Represents a device (extension/station). |
| Oracle.RightNow.Cti.Model.IEmail | Represents an email interaction. |
| Oracle.RightNow.Cti.Model.IInteraction | Represents an interaction. This is the common interface used by all interaction types. |

Enums

|  |  |
| --- | --- |
| Type | Description |
| Oracle.RightNow.Cti.Model.AgentSwitchMode | Represents the possible agent modes on the target switch. |
| Oracle.RightNow.Cti.Model.CallType | Represents the call type. |
| Oracle.RightNow.Cti.Model.InteractionState | Represents the interaction state. |
| Oracle.RightNow.Cti.Model.MediaType | Represents an interaction media type. |
| Oracle.RightNow.Cti.Model.TransferResult | Represents the result of a transfer operation. |
| Oracle.RightNow.Cti.Model.TransferTypes | Represents a transfer type. |

The Extensibility Model

The toolkit uses the Managed Extensibility Framework (MEF) for type composition/DI. MEF is part of the .NET framework, so there are no additional dependencies. An overview of MEF is outside of the scope of this document, but you can find more information on MSDN (<http://msdn.microsoft.com/en-us/library/dd460648(v=vs.100).aspx> ).

The toolkit uses a directory catalog using the deployment directory as the source, so assemblies located is the same directory as the add-in will be used in the composition process.

Imported Types

The core toolkit runtime imports the following types:

|  |  |
| --- | --- |
| Type | Description |
| Oracle.RightNow.Cti.MediaBar.IMediaBarProvider | Provides the media bar view (UI) used by the toolkit. |
| Oracle.RightNow.Cti.Configuration.IConfigurationProvider | Provides configuration information. |
| Oracle.RightNow.Cti.ICredentialsProvider | Provides the logic to resolve agent credentials. |
| Oracle.RightNow.Cti.Model.IInteractionProvider | Provides the switch specific implementation that allows the toolkit to communicate with the target platform. |
| Oracle.RightNow.Cti.Model.IScreenPopHandler | Handles screen pop logic.  The runtime supports many simultaneous handlers and they will all be invoked when a new interaction is delivered. |

Exported Types

The core toolkit runtime exports the following types:

|  |  |
| --- | --- |
| Type | Description |
| Oracle.RightNow.Cti.MediaBar.IGlobalContext | The RightNow Connect Add-In Framework Global context. |
| Oracle.RightNow.Cti.InteractionManager | A type that manages the toolkit’s runtime context. |
| Oracle.RightNow.Cti.CredentialsProvider | A default implementation of Oracle.RightNow.Cti.ICredentialsProvider. This implementation is a simple example that uses the currently logged in user’s information to create a set of credentials. |
| Oracle.RightNow.Cti.Configuration.ConfigurationProvider | A default implementation of Oracle.RightNow.Cti.Configuration.IConfigurationProvider. This implementation uses custom objects in the RightNow platform to persist and retrieve configuration information. |

Any type involved in the composition process may import the types exported by the toolkit.

In addition to the core types, there are also sample implementations of the screen pop, media bar and interaction providers. Those implementations will be covered in the subsequent sections of this document.

Implementing a Provider

The term “provider” is used here in reference to the full set of types that will allow the toolkit to communicate with a new switch. In this context, we are referring to a collection of types. In this section, we will walk through the steps required to create each of the types that a new provider should implement.

Creating the Interaction Provider

The interaction provider is the glue between the toolkit and the switch. It abstracts all of the switch specific details away from the toolkit’s runtime. This is the primary type and the entry point of your switch specific implementation.

In order to create the provider, the first step we need to take is to implement the IInteractionProvider interface.



As you can see, the interface defines a series of properties, methods and events that are used by toolkit through the life of an agent session. A sample provider implementation is available in the companion sample projects and demonstrates how those members are implemented.

As mentioned at the beginning of this section, a complete provider is expected to implement a collection of types, now that we have covered the core type of our new implementation, let’s look at the supporting types required by our implementation.

The Interaction Types

As part of the object model defined by the toolkit, there are a series of interaction types for the different media types supported. All of the supported interaction types implement interfaces that ultimately derive from the common IInteraction interface. Those interaction objects are managed by the interaction provider, and are part of the extensibility model provided by the toolkit.

As mentioned above, there are specific interfaces for each interaction/media type supported by the toolkit. You can also define your own media types and interfaces, as long as they derive from IInteraction.

The built in media specific interaction interfaces are:

|  |  |
| --- | --- |
| Interface | Description |
| ICall | Implemented if the provider supports voice interactions. |
| IEmail | Implemented if the provider supports email interactions. |
| IWebIncident | Implemented if the provider supports web interactions. |

TIP: Since there is a lot of common functionality between the interaction types, it is recommended that all interaction types derive from a common abstract class that implements IInteraction.

The Agent

The agent object (defined by the IAgent interface) abstracts the internal implementation details of how the target platform manages the agent state and contains basic information about the agent using the system.

The Device

Similar to the agent object, the device (defined by the IDevice interface) abstracts the internal implementation details of how the target platform manages the device (extension/station). The toolkit uses the provider’s implementation to manage transfers, DTMF tones, state and event notifications.

Summary

The previous sections outlined the steps required to develop a new interaction provider that enables the Oracle RightNow Multimedia Toolkit to communicate with a new switch/telephony platform. To recap, the steps are:

* Create a new interaction provider by implementing the IInteractionProvider interface.
* Create one or more interaction types for every supported media (by implementing ICall, IEmail, IWebIncident or another IInteraction derived type).
* Create the agent object by implementing the IAgent interface.
* Create the device object by implementing the IDevice interface.

The Sample Provider

The Oracle Service Cloud Multimedia Toolkit is packaged with a sample provider implementation. This provider communicates with a simulated CTI environment (also included in the toolkit’s package) and should serve as a reference implementation.

Screen Pop

Screen pops are an integral part of any CTI solution. To enable a rich screen pop experience, the toolkit defines an extensibility point that makes use of the IScreenPopHandler interface.

Multiple screen pop handlers may be exported and they will all be notified when a new interaction arrives.

Here is a sample screen pop handler implementation (part of the toolkit samples):

[Export(typeof(IScreenPopHandler))]

public class SimpleScreenPopHandler : IScreenPopHandler, IPartImportsSatisfiedNotification {

private RightNowObjectProvider \_objectProvider;

[Import]

public IGlobalContext RightNowGlobalContext { get; set; }

public void HandleInteraction(IInteraction interaction) {

switch (interaction.Type)

{

... Other types removed for brevity. Please refer to the sample implementation for details...

case MediaType.Email:

RightNowGlobalContext.AutomationContext.EditWorkspaceRecord(WorkspaceRecordType.Incident, long.Parse(interaction.AdditionalIdentifiers["ReferenceId"]));

break;

default:

return;

}

}

public void OnImportsSatisfied() {

\_objectProvider = new RightNowObjectProvider(RightNowGlobalContext);

}

}

Known Issues

* The current implementation does not support chat. Chat support is planned and will be addressed in a future release.