Creatio Academy

Telephony integration

Integration with Oktell

Version 8.0

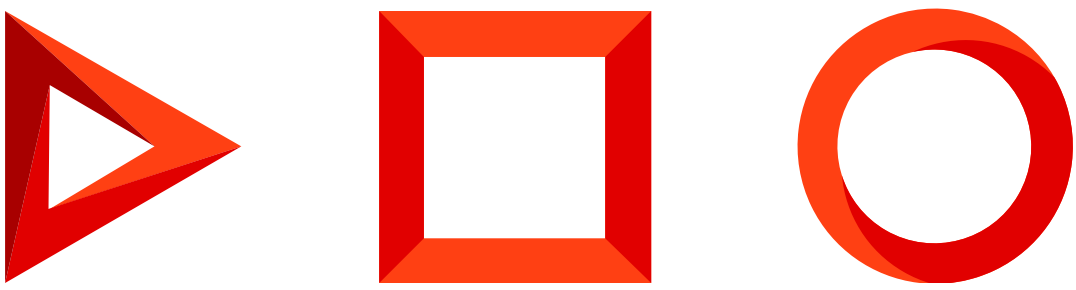


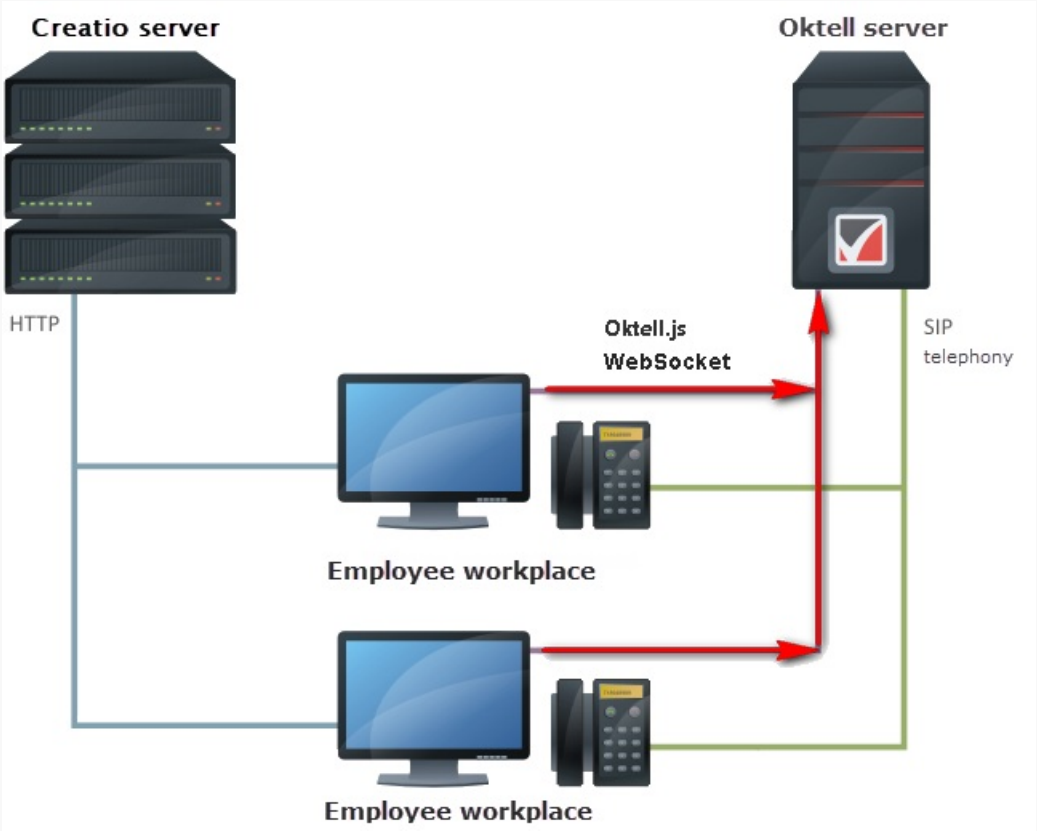
Table of Contents

Integration with Oktell

Advanced

Oktell integration with Creatio is implemented on the client level using the oktell.js library. The oktell.js source code is located in the `OktellModule` configuration schema of the `CTIBase` package.

The Oktell server communicates with phones and with the end clients (browsers). With this integration method Creatio does not requires its own WebSocket server. Each client connects via the WebSocket Protocol directly to the Oktell server. The Creatio application server creates pages and provides data from the application database. There is no direct relationship between Creatio and Oktell server. Access is not required, so customers process and combine the data of the two systems independently. The Oktell web client and the oktell.js plugin, embedded in other projects, are implemented according to this principle.



Oktell.js

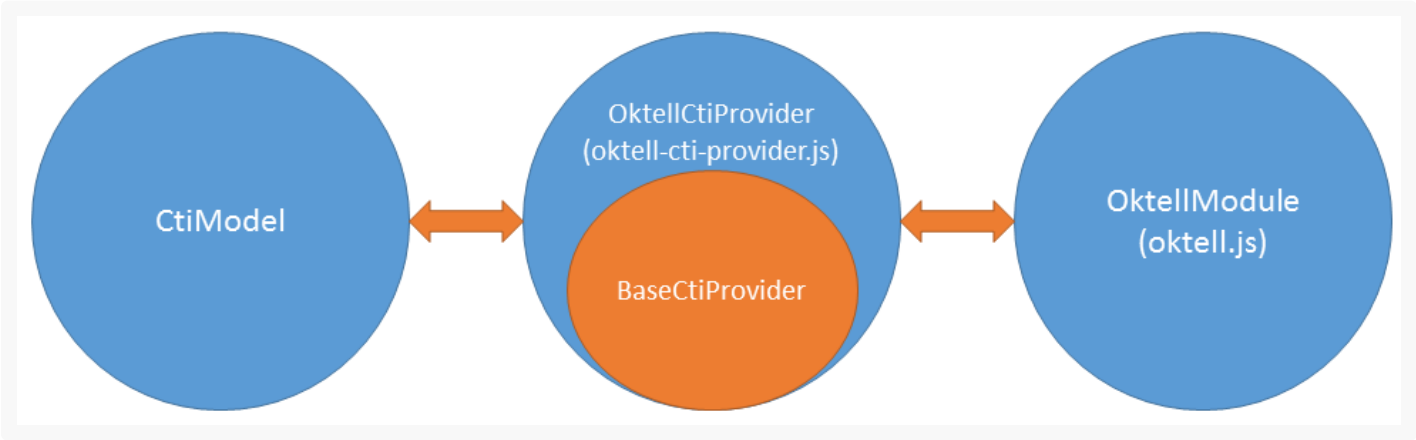
Oktell.js is a javascript library for embedding the functionality of the call control in a CRM system. Oktell.js uses the Oktell WebSocket Protocol to connect to the Oktell server. The advantage of this Protocol is the establishing of a permanent asynchronous connection to the server, which enables you to receive events from the server Oktell and execute certain commands. Because the Oktell WebSocket protocol is quite complicated to implement, the Oktell.js wraps the WebSocket Protocol methods inside itself thus providing simple management functionality.

Voice transmission between subscribers

In a conversation between the oktell and Creatio operators, voice is transmitted via the Session Initiation Protocol (SIP). This requires that either the VoIP phone or the Softphone operator be installed on your computer.

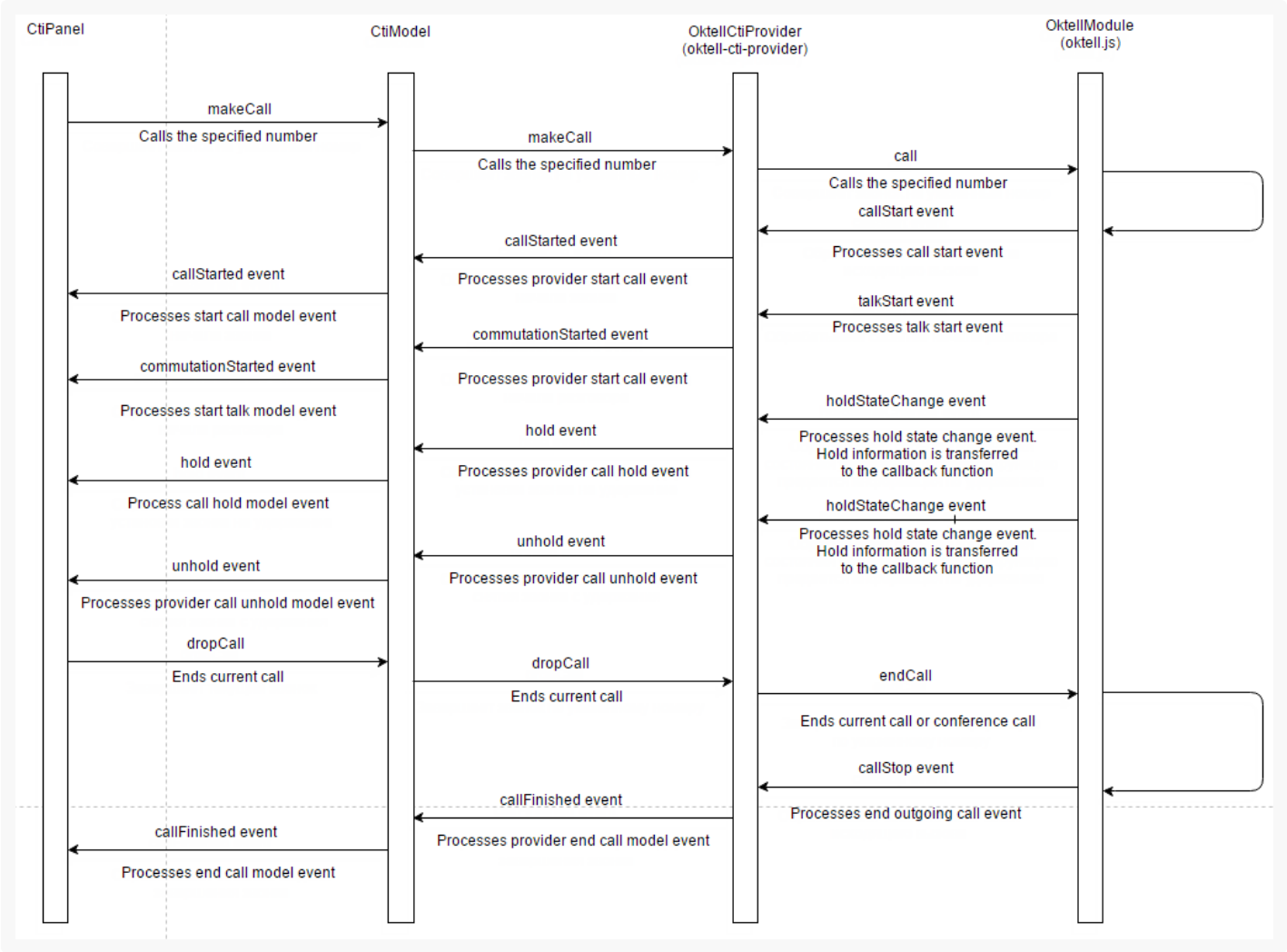
Interaction of components

The interaction with the oktell.js library is executed via the *OktellCtiProvider* class, which is a link between *CtiModel* and *OktellModule* that contains the oktell.js code. The `OktellCtiProvider` class implements the `BaseCtiProvider` interface class.

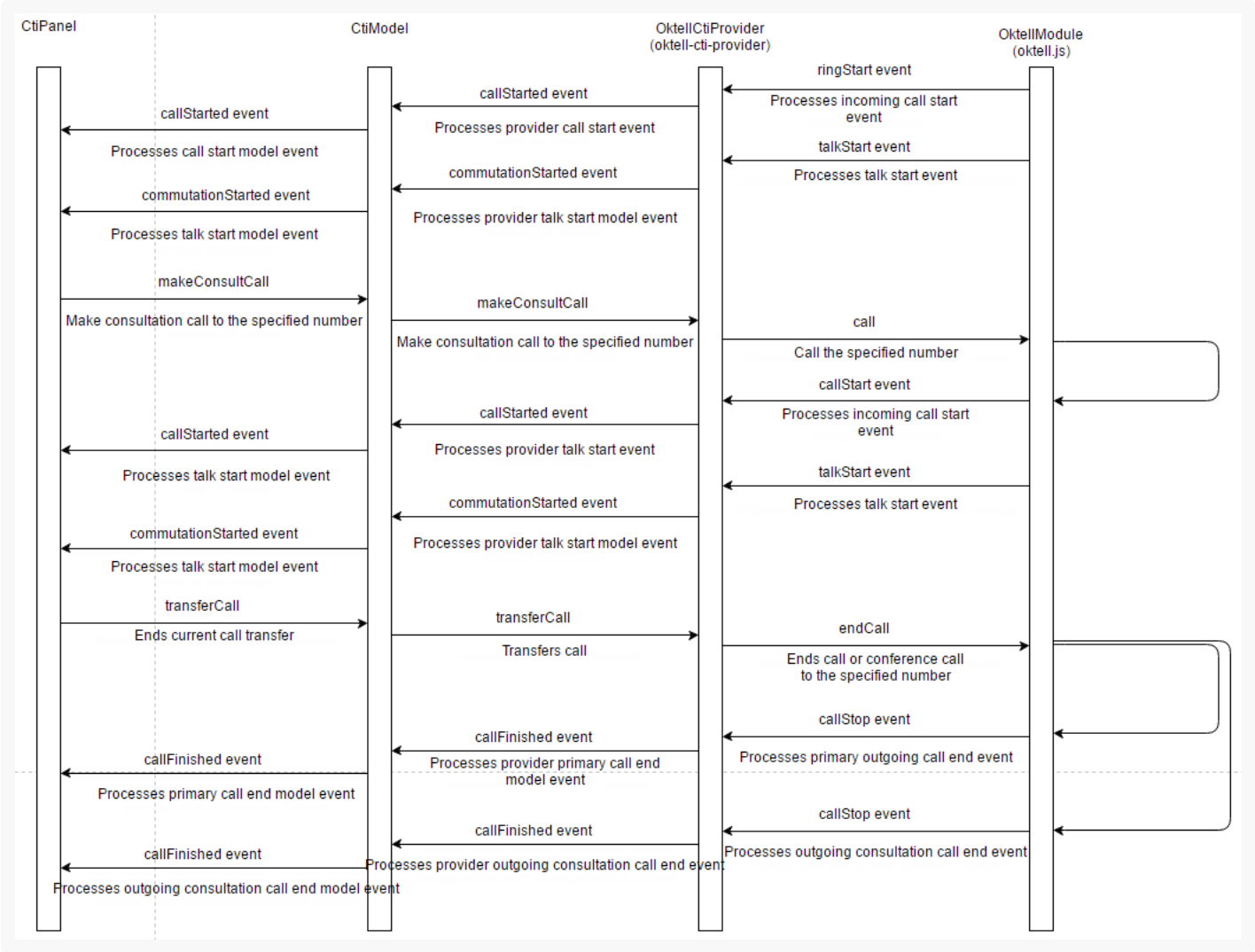


Examples of interaction between CtiModel, OktellCtiProvider and OktlellModule:

Operator outgoing call to a subscriber: putting a call on hold, putting off hold by a subscriber and finishing the call by the operator



Incoming call of a subscriber 1 to an operator with a consultation call to subscriber 2 with the subsequent connection of the subscriber 1 and subscriber 2 by the operator



The list of supported oktell.js class library events is listed in table.

The list of supported oktell.js class library events

Event	Description
`connect`	Successful connection to server event.
`connectError`	Connection to server error in the `connect` method event. Error codes are the same as for the callback function of the `connect` method.
`disconnect`	Server connection closing event. The object describing the reason of the disconnection is passed to the callback function.
`statusChange`	Agent status change event. Two string parameters are passed to the callback function — the new and previous state.
`ringStart`	Incoming call start event.
`ringStop`	Incoming call stop event.
`backRingStart`	Returning call start event.
`backRingStop`	Returning call stop event.
`callStart`	Outgoing call start event.
`callStop`	Call UUID change event.
`talkStart`	Conversation start event.
`talkStop`	Conversation stop event.
`holdAbonentLeave`	Caller hold leave event The `abonent` object is passed to the callback function with information on the caller.
`holdAbonentEnter`	Caller hold enter event The `abonent` object is passed to the callback function with information on the caller.
`holdStateChange`	Hold status change event. The information on the hold is passed to the hold function.
`stateChange`	Line status change event.
`abonentsChange`	Current abonents list change event.
`flashstatechanged`	Hold status change low-level event.
`userstatechanged`	User status change low-level event.