Neocase®

Case Form JavaScript Framework

Technical Guide

TECHNICAL DOCUMENTATION

REFERENCE

Document | NeoPW_Tec_CaseFormJSFrwk_15

Version | Version #1

Product | Neocase

Issue Date | Thursday, January 2, 2019

COPYRIGHT

PUBLISHER

Neocase Software SAS
P: +33 (0)1 73 60 01 10 | F: +33 (0)1 80 04 84 99
Americas HQ: 275 Grove St. Suite 2-400, Newton, MA 02466 - USA
Europe HQ: 91 Boulevard National, 92250 La Garenne-Colombes - FRANCE
Northern Europe: 2 Kingdom Street, Paddington Basin, London W2 6BD - UK
www.neocasesoftware.com

Content

Document Version

VERSION	DESCRIPTION	DATE
1	Document creation	26/06/2015

1 | Overview

This document covers the JavaScript Framework of the Case Form offered by Neocase POWER.

Warning! The standard JavaScript code does not always apply to Neocase.

1.1 | Prerequisites for Use of this Document

Before reading this guide, you should have a basic knowledge of JavaScript language.

For more technical details regarding Neocase POWER, please contact our support. We will be happy to provide you with more technical information and documentation.

1.2 | Document Version Compliance

The version of this document is compliant with Neocase Power 2012 & 14.

2 | Fields

2.1 | Form Fields

The form fields contain case values and can be used in JavaScript functions. They are categorized in 3 parts.

2.1.1 | Property Fields

The best way for accessing some fields value is to use the following javascript method.

For example, for the user email: document.getElementById("UTILISATEURS$EMAIL_UTILISATEUR").value

Field name	Type	Description
INTERVENTIONS_EN_COURS$NUMERO	Number	Case number
INTERVENTIONS_EN_COURS$CODEINTERVENANT	String	Agent code
INTERVENTIONS_EN_COURS$DATEQUESTION	Date	Date of creation
INTERVENTIONS_EN_COURS$QUESTION_INTERVENTION	String	Question
INTERVENTION_EN_COURS$CATEGORIE	Number	Category code
INTERVENTION_EN_COURS$TYPE	Number	Type code
INTERVENTION_EN_COURS$ELEMENT	Number	Item code
INTERVENTION_EN_COURS$MOTCLE	Number	Keyword code
INTERVENTION_EN_COURS$ENVIRONNEMENT	Number	Environment code
INTERVENTION_EN_COURS$ETAT	Number	Status code
INTERVENTION_EN_COURS$PROVENANCE	Number	Origin code
INTERVENTION_EN_COURS$FOURNISSEUR	Number	Sub-contractor code
INTERVENTIONS_EN_COURS$DELAI	String	Severity
INTERVENTIONS_EN_COURS$CODECARTE	Number	Card code
INTERVENTIONS_EN_COURS$DATE_ECHEANCE	DateTime	Deadline date
INTERVENTIONS_EN_COURS$REPONSE_REPONSE	String	Response
INTERVENTIONS_EN_COURS$NOM_FILE	String	Queue name
INTERVENTIONS_EN_COURS$SLA	String	SLA label
INTERVENTIONS_EN_COURS$VALEUR1 **Note -** The value can be comprised between 1 and 999.	String	Custom field

Field name in the Back Office	Field name in the Portal	Type	Description
INTERVENTION_EN_COURS$CATEGORIE	CATEGORIES	Number	Category code
INTERVENTION_EN_COURS$TYPE	TYPES	Number	Type code
INTERVENTION_EN_COURS$ELEMENT	ELEMENTS	Number	Item code
INTERVENTION_EN_COURS$MOTCLE	MOTCLES	Number	Keyword code
INTERVENTION_EN_COURS$ENVIRONNEMENT	ENVIRONMENTS	Number	Environment code
INTERVENTION_EN_COURS$ETAT	ETATS	Number	Status code
INTERVENTION_EN_COURS$PROVENANCE	PROVENANCES	Number	Origin code
INTERVENTION_EN_COURS$FOURNISSEUR	FOURNISSEURS	Number	Sub-contractor code
INTERVENTIONS_EN_COURS$DELAI	DELAIS	String	Severity

2.1.2 | Contact Fields

Field name	Type	Description
UTILISATEURS$CONTACT_INTERVENTION	String	Interlocutor name
UTILISATEURS$MATRICULE_UTILISATEUR	Number	Contact code
UTILISATEURS$GENRE_UTILISATEUR	String	Contact type
UTILISATEURS$NOM_UTILISATEUR	String	Contact last name
UTILISATEURS$PRENOM_UTILISATEUR	String	Contact first name
UTILISATEURS$TELEPHONE_UTILISATEUR	String	Contact phone
UTILISATEURS$TELECOPIE_UTILISATEUR	String	Contact fax
UTILISATEURS$EMAIL_UTILISATEUR	String	Contact email
UTILISATEURS$ADRESSE1_UTILISATEUR	String	Contact address 1
UTILISATEURS$ADRESSE2_UTILISATEUR	String	Contact address 2
UTILISATEURS$CP_UTILISATEUR	String	Contact zip code
UTILISATEURS$VILLE_UTILISATEUR	String	Contact city
UTILISATEURS$REGION	String	Contact state
UTILISATEURS$PAYS_UTILISATEUR	Number	Contact country
UTILISATEURS$CHAMPU1 **Note -** The value can be comprised between 1 and 999.	String	Contact custom field 1 to 20

2.1.3 | Contract Fields

Field name	Type	Description
ABONNEMENTS$CARTE	String	Card label
ABONNEMENTS$CODEABONNEMENT	Number	Contract code
ABONNEMENTS$ABONNEMENT	String	Contract label
ABONNEMENTS$TYPE_ABONNEMENT	String	Contract type
ABONNEMENTS$DATE_DEBUT	Date	Contract start date
ABONNEMENTS$DATE_FIN	Date	Contract expiration date
ABONNEMENTS$CHAMPA1 **Note -** The value can be comprised between 1 and 999.	String	Contract custom field 1 to 20

2.2 | Active Fields

The active fields are information extracted from cases. They can be used when calling out an application using a URL, an iFrame or a shortcut.

Field name	Type	Description
{#number#}	{#numero#}	Case number
{#parentnumber#}	{#numeroparent#}	Parent case number
{#date#}	{#date#}	Creation date of the case
{#contact#}	{#contact#}	First name and last name of the contact
{#account#}	{#compte#}	Name of the account
{#operator#}	{#intervenant#}	Name of the agent
{#team#}	{#equipe#}	Name of the agent team
{#queue#}	{#file#}	Name of the queue
{#configurationcode#}	{#codeconfiguration#}	Asset ID
{#contactcode#}	{#codecontact#}	Contact ID
{#conditioncode#}	{#codecondition#}	Service option ID
{#accountcode#}	{#codecompte#}	Account ID
{#rootaccountcode#}	{#codecompteracine#}	Root account ID
{#operatorcode#}	{#codeintervenant#}	Agent ID
{#cardcode#}	{#codecarte#}	Card ID
{#contractcode#}	{#codecontrat#}	Contract ID

Field name	Type	Description
{#modecreation#}	{#modecreation#}	Boolean string that indicate the creation mode
{#securityid#}	-	Session GUID
{#myoperatorcode#}	-	ID of the connected agent
{#operatorcode#}	-	ID of the case agent
{#processingcontact#}	-	Full name of the processing delegated contact
{#othercontact#}	-	Full name of the requestor

Note - Both English and French names are recognized by the system disregarding the language of the user interface

3 | Case Form

Note - All these properties are available in the BackOffice

3.1 | Case Object

The case object is the most important object of the case form; it allows accessing the case entity with reading or modifying rights.

3.1.1 | This Case

Properties

Name	Type	Description	Syntax
`ServiceOptionID`	Numeric	Get the ID of the service option used for the case	`ThisCase.ServiceOptionID`
`CardID`	Numeric	Get the ID of the card associated with the case.	`ThisCase.CardID`
`ConfigurationID`	Numeric	Get the ID of the configuration associated with the case	`ThisCase.ConfigurationID`
`ContactID`	Numeric	Get the ID of the contact associated with the case	`ThisCase.ContactID`
`ContactBisID`	Numeric	Get the ID of the secondary contact associated with the case	`ThisCase.ContactBisID`
`ContactProcessingID`	Numeric	Get the ID of the contact who's delegated to update the case from enterprise portal	`ThisCase.ContactProcessingID`
`AccountID`	Numeric	Get the ID of the account associated with the case	`ThisCase. AccountID`
`QueueID`	Numeric	Get the ID of the processing queue in which the case was placed	`ThisCase.QueueID`
`IsClosed`	Boolean	Indicate if the case is closed or not closed	`ThisCase.IsClosed`
`IsNewCase`	Boolean	Indicate if the case is in creation or modification mode	`ThisCase.IsNewCase`

Name	Type	Description	Syntax
`IsSubContracted`	Boolean	Indicate if the case is subcontracted	`ThisCase.IsSubContracted`
`IsVisibleByContact`	Boolean	Indicate if the case is visible by contact	`ThisCase.IsVisibleByContact`

Methods

Name	Description	Syntax
`ExecuteRules(N1, N2, …)`	Execute the specified rules N1 = name of the first rule N2 = name of the second rule **Note -** Only manual rules can be executed by this method.	`ThisCase.ExecuteRules('RULE 1', 'RULE 2')`
`BackgroundMode.Begin()`	Start background mode execution	`ThisCase.BackgroundMode.Begin()`
`BackgroundMode.Execute(F)`	Execute an action in background mode F = function name that represent an action (string)	`ThisCase.BackgroundMode.Execute ('enregistreronly()')`
`BackgroundMode.End(C)`	End background mode and execute the callback code. C = callback function (string or function)	`ThisCase.BackgroundMode.End ('envoiReponseParMail()')`
`BackgroundMode.Stop()`	Stop the background mode	`ThisCase.BackgroundMode.Stop()`
`GetProperty(P, A)`	Get a case property dropdown list value or text P = Property dropdown list name (string) A = Optional attribute ('value' or 'text') by default 'value'	`ThisCase.GetProperty ('INTERVENTIONS_EN_COURS$TYPE')`
`SetProperty(P, V)`	Set a case property dropdown list value	`ThisCase.SetProperty ('INTERVENTIONS_ EN_`

Name	Description	Syntax
	P = Property dropdown list name (string) V = Value to set (Numeric)	`COURS$TYPE',0)`
`SetQuestion(T)`	Set the current question T = text of the question	`ThisCase.SetQuestion ("My question")`
`SetNewResponse(T)`	Set the new response T = text of the new response	`ThisCase. SetNewResponse ("My response")`
`SetCumulativeResponse(T)`	Set the cumulative response got from the history	`ThisCase. SetCumulativeResponse ("My cumulative response")`

`GetDeadline()`	Get the deadline object
	Object.Id = The severity identifier (number)
	Object.Severity = The severity name (string)
	Object.Duedate = The datetime in french format (dd/mm/yyyy hh24:mi)
		`var Object = ThisCase.GetDeadline()`

`SetDeadline (severityName, dueDate)`	Set case deadline date time severityName = The name of the severity (string) dueDate = The date time in French format (dd-mm-yyyy hh24:mi:ss)	`ThisCase.SetDeadline ("30 jours","22-03-2017 13:30:00")`
`GetNewDeadline (startDate, startHM, period, periodUnit, timeTableID)`	Calculate deadline working date time from start date startDate = The start date in French format (dd/mm/yyyy) startHM = The start hour and minutes (hh24:mi)	`var DueDate = ThisCase.GetNewDeadline ("28/01/2011","14:36", 3, "j", 0)`

Name	Description	Syntax
	period = amount of working period (numeric) periodUnit = period unit (j = days or s = seconds) timeTableID = The time table identifier	
`GetNewDeadlineDay(startDate, startHM, period, timeTableID)`	Calculate deadline working date time from start date startDate = The start date in French format (dd/mm/yyyy) startHM = The start hour and minutes (hh24:mi) period = amount of working days timeTableID = The time table identifier	`var Days = ThisCase.GetNewDeadlineDay (28/01/2011","14:36", 3, 0)`
`GetNewDeadlineSecond (startDate, startHM, period, timeTableID)`	Calculate the deadline working date time from start date startDate = The start date in French format (dd/mm/yyyy) startHM = The start hour and minutes (hh24:mi) period = amount of working seconds timeTableID = The time table identifier	`var Seconds = ThisCase.GetNewDeadlineSecond ("28/01/2011","14:36", 3000, 0)`
`GetNewStartDate (dueDate, dueHM, period, periodUnit,timeTableID)`	Calculate the start working date time from due date. dueDate = The due date in French format (dd/mm/yyyy) dueHM = The due date hour and minutes (hh24:mi) period = amount of time period periodUnit =	`var StartDate = ThisCase.GetNewStartDate ("07/02/2011","14:00", 3, "J",0)`

Name	Description	Syntax
	working time period unit (J = working days, S = working seconds, j = absolute days, s = absolute secondes) timeTableID = The time table identifier	
`GetNewStartDateDay (dueDate, dueHM, period, timeTableID)`	Calculate the start working date time from due date. dueDate = The due date in French format (dd/mm/yyyy) dueHM = The due date hour and minutes (hh24:mi) period = amount of working days timeTableID = The time table identifier	`var StartDate = ThisCase.GetNewStartDateDay ("07/02/2011","14:00", 3,0)`
`GetNewStartDateSecond (dueDate, dueHM, period, timeTableID)`	Calculate the start working date time from due date. dueDate = The due date in French format (dd/mm/yyyy) dueHM = The due date hour and minutes (hh24:mi) period = amount of working seconds timeTableID = The time table identifier	`var StartDate = ThisCase.GetNewStartDateSecond ("07/02/2011","14:00", 60,0)`
`AddRequiedFieldBeforeAction (actionName, fieldName, alertMessage, initVal)`	Add field to validate change before executing an action actionName = The name of an action (string) fieldName = The field name (string) alertMessage = The alert message (string) initVal = The initial value to compare to (string)	`AddRequiedFieldBeforeAction ("save","INTERVENTIONS_ EN_ COURS$VALEUR1","Required field","")`

Name	Description	Syntax
InsertJSBeforeAction (actionName, JavaScript)	Insert JavaScript code to execute before an action	InsertJSBeforeAction ("save", "alert('hello');")

AddNotificationAddress (address, gateway)	Add email address in the case notification list
	address = Email address
	gateway = The gate way (1 = EMAIL, 2 = SMS, 3 = FAX)
		AddNotificationAddress ("support@neocase.com", 1)

Save()	Save the case information and stay on form window	ThisCase.Save()
SaveAndClose()	Save the case information and close the form window	ThisCase.SaveAndClose()
Transfer()	Open the case transfer to an agent, a team, in queue dialog	ThisCase.Transfer()
TransferBySkill()	Open the case transfer by skill dialog	ThisCase.TransferBySkill()
TransferByLocalisation()	Open the case transfer by localization dialog	ThisCase.TransferByLocalisation ()
SubContract()	Open the case subcontracting dialog	ThisCase.SubContract()
DelegateToContact()	Open the case delegation contact dialog	ThisCase.DelegateToContact()
Resolve()	Open the case resolution dialog	ThisCase.Resolve()
Close()	Open the case closure dialog	ThisCase.Close()
Cancel()	Cancel the modifications and close form window	ThisCase.Cancel()
CreateChild(cardID, configurationID, contactID,	Create child case cardID = The	ThisCase. CreateChild (n,n,n,n,n, 'SERVICE OPTION')

Name	Description	Syntax
accountID, queueID, serviceOptionName)	associated card identifier (null = none) configurationID = The associated configuration identifier (null = none) contactID = The associated contact identifier (null=none) accountID = The associated account identifier (null = none) queueID = The queue identifier (null = default queue) serviceOptionName = The service option name (null = same as parent)	
CreateDocument (docID, docName, bAttach)	Create a document from case docID = the document identifier docName = The document name bAttach = Option to attach document with the case	ThisCase.CreateDocument (n, 'NAME', false)
CreateScheduling (subject, description, startdatetime, enddatetime)	Create scheduling for case subject = subject (string) description = description (string) startdatetime = start date time in French format "dd/mm/yyyy hh24:mi" (string) enddatetime = end date time in French format "dd/mm/yyyy hh24:mi" (string)	ThisCase.CreateScheduling ('subject', 'description', '01/02/2011 12:00', '01/02/2011 12:30')

Associated Entities

ThisCase.Agent

Properties

Name	Type	Description	Syntax
`Id`	Numeric	Get the case agent ID	`ThisCase.Agent.Id`
`FirstName`	String	Get the agent first name	`ThisCase.Agent.FirstName`
`LastName`	String	Get the agent last name	`ThisCase.Agent.LastName`
`TeamID`	Numeric	The agent team identifier	`ThisCase.Agent.TeamID`

Methods

Name	Description	Syntax
`GetField(N)`	Get the value of the field from database N= The field name	`ThisCase.Agent.GetField ("CODELANGUE")`

ThisCase.Contact

Properties

Name	Type	Description	Syntax
`Id`	Numeric	Get the associated contact ID	`ThisCase.Contact.Id`
`FirstName`	String	Get the associated contact first name	`ThisCase.Contact.FirstName`
`LastName`	String	Get the associated contact last name	`ThisCase.Contact.LastName`

Methods

Name	Description	Syntax
`GetField(N)`	Get the value of the field from database N= The field name	`ThisCase.Contact.GetField("CODESLA")`

ThisCase.Account

Properties

Name	Type	Description
`Id`	Numeric	`Get the associated account ID`
`Name`	String	`Get the associate account name`

Methods

Name	Description	Syntax
`GetField (N)`	Get the value of the field from database N= The field name	`ThisCase.Account.GetField ("PAYS_STRUCTURE")`

ThisCase.Configuration

Properties

Name	Type	Description
`Id`	Numeric	`Get the associated configuration ID`

Methods

Name	Description	Syntax
`GetField (N)`	Get the value of the field from database N= The field name	`ThisCase.Configuration.GetField ("CONFIGURATION")`

ThisCase.Contract

Properties

Name	Type	Description
`Id`	Numeric	Get the associated contract ID
`Card`	Object	Get the associated card of the contract
`Card.Id`	Numeric	Get the associated card ID

Name	Description	Syntax
`GetField(N)`	Get the value of the contract field from database N= The contract field name	`ThisCase.Contract.GetField ("ABONNEMENT")`
`Card.GetField (N)`	Get the value of the card field from database N= The card field name	`ThisCase.Contract.Card.GetField ("TYPECARTE")`

3.2 | Form Object

The form object is used to manipulate the interface of the case form.

3.2.1 | This Form

Events

Name	Description	Syntax	Available in the Enterprise Portal
`OnLoadComplete`	Execute JavaScript code after loading is complete	`ThisForm.OnLoadComplete = "Function()"`	Yes
`OnSubmit`	Execute JavaScript code on form submitting	`ThisForm.OnSubmit = "Function()"`	Yes
`ThisForm.Bind`	Abonnement à un événement	`ThisForm.Bind (“loadcomplete”,Function)`	Yes
`ThisForm.Unbind`	Désabonnement d'un événement	`ThisForm.Unbind (“loadcomplete”)`	Yes
`ThisForm.Trigger`	Déclenchement d'un événement	`ThisForm.Trigger (“loadcomplete”)`	Yes

Methods

Name	Description	Syntax	Available in the Enterprise Portal
`GetElement (Name)`	Get a HTML element by name	`Var Object = ThisForm.GetElement ("INTERVENTIONS_EN_COURS$TYPE")`	Yes
`HideSection (sectionId)`	Hide a visible section	`ThisForm.HideSection("section1")`	Yes
`ShowSection (sectionId)`	Show a hidden section	`ThisForm.ShowSection("section1")`	Yes

UI Objects

ThisForm.TabPane

This class represents the tab object.

Properties

Name	Type	Description	Syntax
`selectedIndex`	Numeric	Index of the current tab	`ThisForm.TabPane.selectedIndex`
`pages[I]`	Object	Get the tab page by index I = index of the page started by 0	`ThisForm.TabPane.pages[0]`
`pages [I].onselect`	Function	Set/Get onselect event on tab page	`ThisForm.TabPane.pages [0].onselect = function(){…}`

Methods

Name	Description
`Select(I)`	Select the tab by index I = index of the tab started by 0
`Hide(I)`	Hide the tab by index I = index of the tab started by 0
`Show(I)`	Show the tab by index I = index of the tab started by 0

ThisForm.IframeProperties

This class represents the Iframe that is used to load the property fields values.

Events

Name	Description	Syntax
`OnLoadComplete (code)`	Execute code after property fields values are completely loaded	`ThisForm.PropertiesIframe.OnLoadComplete (code)`

3.3 | Session Object

The Session object allows getting information about running application and the connected agent.

3.3.1 | This Session

Properties

Name	Type	Description	Syntax
`Language`	String (ISO CODE)	Get the current session language code	`ThisSession.Language`
`TeamScope`	Numeric	Get the current team scope code	`ThisSession.TeamScope`

Name	Type	Description	Syntax
`IsContractEnabled`	Boolean	Is contract module is enabled	`ThisSession.IsContractEnabled`
`IsAssetEnabled`	Boolean	Is asset module is enabled	`ThisSession.IsAssetEnabled`

3.3.2 | ThisSession.Agent

Properties

Name	Type	Description	Syntax
`Id`	Numeric	Get the connected agent unique identifier	`ThisSession.Agent.Id`
`FirstName`	String	Get the connected agent first name	`ThisSession.Agent.FirstName`
`LastName`	String	Get the connected agent last name	`ThisSession.Agent.LastName`
`TeamID`	String	Get the connected agent team unique identifier	`ThisSession.Agent.TeamID`

Methods

Name	Description	Syntax
`GetField(N)`	Get the field value from database	`ThisSession.Agent.GetField("CODECULTURE")`

3.4 | JavaScript Functions

These functions can be called from any button, shortcut or fields in the form.

Warning! These functions are likely to change. Please, ensure that there are up to date before using them.

Note - These functions are available from the Back Office only.

Function	Description
afficherFichierLie()	Open the attachment dialog box
afficherHistoriqueInterventions ()	Display the history of the cases for this contact, its account, the root account or the asset
afficherInformationsAvancees ()	Display the advanced information in a new window for the contact or the account

Function	Description
afficherMail()	Display the history of mail
afficherOperationsActions()	Display the history of operations for this case
afficherPostes()	Display the asset assigned to the account or the contact for choosing or changing the asset
affilierinterventionparent()	Assign the case to a parent case
annuler()	Exit from the case without taking account the modifications
appliquerModeles()	Display the list of question and answer templates
basculer()	Transfer the case to an agent, a team or a queue
basculerParCompetence()	Transfer the case to an agent using his skill, use the value of the current selected element
changerCarte()	Change the card
changerCondition()	Change the service option
changerStructure()	Change the account
changerUtilisateur()	Change the contact
changerUtilisateur('bis')	Change the requestor
cloturer()	Close a case
consulterScenarios()	Display the list of scripts
correctionOrthographique()	Run the spellchecker
creerDocument()	Generate a document
creerDocumentparticulier()	Generate a document using a template which name is provided as parameter. Parameters are the following: document template name and id of the document template (MODELES_TEMPLATE_DOC table) separated by a comma.
creerintervention()	Create a case
creerinterventionfille()	Create a child case
creerUtilisateur()	Create a new contact
envoiReponseParMail()	Send a follow-up email
escalader()	Transfer outside case to a sub-contractor
executerRegles()	Execute one or several rules. Rule names should be between quotes ' and rules are separated by columns.
ouvrirAide()	Open the on line help
ouvririntervention()	Open a case
ouvririnterventionparent()	Open a parent case

Function	Description
resoudre()	Solve a case
visualiserInterventionFilles()	Display a child case
voirBaseExperience()	Display the search form among the experience base
creerPlanification (subjet, description, startdate, enddate)	Create new event with specified parameters. The syntax is the following: `creerPlanification (Event subject, Event description, DD/MM/YYYY hh24:mi :ss, DD/MM/YYYY hh24:mi :ss)` **Note -** The date format is the local date format..
ouvrirPlanification()	Open list of events linked to the case

Note - In Javascript, function names are case sensitive. For the execution of a rule in a background, use the following functions:

- ThisCase.BackgroundMode.Begin();
- ThisCase.BackgroundMode.Execute("executerRegles('RULENAME1', 'RULENAME2')");
- ThisCase.BackgroundMode.End("alert('Success !')");