JavaScript Schnittstellenbeschreibung und Beschreibung der SQL-Prozeduren für ttFrame 4

Inhaltsverzeichnis

1 Einleitung

Im Rahmen des *ttFrame 4* und der Verwendung der *ttFramework Technologie* werden für verschiedenen Szenarien (z.B. eingehender Inbound-/Outboundanruf) Verweise auf Webseiten hinterlegt, welche vom Nutzer entsprechend den Projektvorgaben individuell gestaltet werden können.

1.1 Darstellung

Innerhalb des vorliegenden Dokuments kann es verschiedene Pfadangaben, Dokumentenangaben und Verweise geben. Diese werden wie folgt dargestellt:

- Installationspfade/Dateipfade, Verweise und Objektbezeichnung werden *kursiv*, mit Anführungszeichen *„“* dargestellt

- Beispiele (z.B. Dateinamen, Mandantennamen etc.) werden in `Courier` dargestellt

2 Syntax

Obwohl in *Javascript* keine strenge Typendeklaration existiert, werden für die Funktionen die jeweiligen Typen mit angegeben. Sofern es sich um den Typ *„object“* handelt, ist damit gemeint, dass der jeweilige Typ vom Funktionsaufruf abhängig ist und variieren kann.

3 Vorbemerkung

In der Version *ttFrame 4* werden jetzt auch *Extensions* unterstützt, deshalb muss die Schnittstelle mit einem *Javascript-Befehl* initialisiert werden.

Das Einbinden oder Laden eines speziellen *Schnittstellen-Scripts* ist nicht mehr notwendig.

4 Konzeptionelle Änderungen gegenüber vorherigen Versionen

Als wesentliche Änderung unterstützt die *ttFrame-Javascript-Schnittstelle* ab dieser Version die Integration eigener *Provider (Extensions)*.

Bei *Extensions* handelt es sich um Erweiterungen für einzelne Bestandteile (z.B. Kampagnen, Maske), mit denen Hilfe sich deren Funktionsumfang durch die serverseitige Ausführung eigenen Codes erweitern lässt *(s. ttDeveloper-Handbuch)*. Des Weiteren können Funktionen mittels entsprechender Tags *(ScriptVisible)* für die Verwendung in *Javascript* freigegeben und Scripts durch die Einbindung als Ressource bereitgestellt werden, die automatisch nachgeladen werden und ggf. die entsprechenden Funktionen clientseitig aufrufen.

Dieses bietet nicht nur die Möglichkeit, die *ttFrame-Javascript-Schnittstelle* nach Belieben zu erweitern, es können ganze Funktionalitäten (z.B. Leitfäden, Bankleitzahlenprüfung etc.) in *Extensions* modular gebündelt werden, die sowohl den serverseitigen *(C# oder VB.NET)*, als auch den clientseitigen *(JavaScript)* Code in Gänze enthalten.

Im Idealfall wäre hierdurch die generelle Auslieferung von *Javascript-Code* über die Provider und der modalere Aufbau von Masken durch das Auswählen von Komponenten *(Extensions)* in der Administrationsoberfläche (z.B. der Kampagne) denkbar.

5 Implementierung der Schnittstelle

Damit eventuell vorhandene Scripts innerhalb des *iFrames* nachgeladen werden, muss die Schnittstelle durch einen *Javascript-Befehl* initialisiert werden. Dem Befehl wird eine CallBack-Funktion übergeben, die aufgerufen wird, nachdem sämtliche *Extension-Scripts* der Maske geladen und ggf. ebenfalls initialisiert wurden. Der Aufruf sollte im *„onload“-Event* der Seite erfolgen.

ACHTUNG: Die Ausführung von eigenem Initialisierungs-Code sollte nicht im „onload“-Event der Seite, sondern erst nach dem Aufruf der CallBack-Funktion oder in dieser selbst erfolgen, da hiermit sichergestellt ist, dass sämtliche Zusatzkomponenten geladen wurden.

6 Beispiele

6.1 Event für das Laden der Seite abonnieren

Initialisierung der Schnittstelle:

```
//Variabel für das Content-Interface
var contentInterface = null;

window.addEventListener('load', function () {
    //Aufruf der Initialisierung
    this.parent.contentInterface.initialize(window,
        //Abschluss der Initialisierung
        function onInitialized(ci) {
            contentInterface = ci;
            //ggf. eigenen Initialisierungs-Code aufrufen
            contentInterface.addTransferSource('Hotline','0800…', 2);
        },
        //Fehler bei der Initialisierung
        function onInitializeError(e) {
            alert('Initialize contentInterface failed: ' + e.message);
        }
    );
})
```

6.2 Implementierung der Schnittstelle

Die Funktionen des Schnittstellen-Objektes können anschließend direkt über die entsprechend vorgesehene Variable angesprochen werden:

```
function button_onClick() {
    contentInterface.terminateCall('Termcode');
}
```

7 Objekte

Die hier aufgelisteten Objekte sind teilweise - soweit gekennzeichnet - Serialisierungen aus dem *Provider.Interface.V1* bzw. dem *ACD.Interface.V1*. Alle Eigenschaften, die dort mit dem Attribut *[ScriptVisible]* gekennzeichnet sind, stehen im *Content-Interface* zur Verfügung.

user

```
user: {
    id (id des Agenten)
    login (login des Agenten)
    userID (identisch mit id)
    firstName (Vorname des Agenten)
    lastName (Nachname des Agenten)
    workstation (Arbeitsplatz des Agenten)
}
```

Repräsentiert das *User-Objekt* mit den Eigenschaften: *id*, *login*, *userID*, *firstName*, *lastName* und *workstation*.

calljob

```
callJob
```

Repräsentiert den *CallJob (s. ACD.Interface.V1)* aufgrund dessen die Maske aufgerufen wurde.

sessionID

```
sessionId
```

Schnellzugriff für *callJob.SessionId (s. ACD.Interface.V1)*

customer

```
customer
```

Schnellzugriff für *callJob.Customer (s. ACD.Interface.V1)*

campaign

```
campaign
```

Schnellzugriff für *callJob.Campaign (s. ACD.Interface.V1)*

mask

```
mask
```

Schnellzugriff für *callJob.Campaign.Mask (s. ACD.Interface.V1)*

extensions

```
extensions
```

Schnellzugriff für *callJob.Campaign.Mask.Extensions (s. ACD.Interface.V1)*

ACHTUNG: Die Extension-Auflistung muss initial mittels getValue() abgerufen werden. Dieses sollte innerhalb der Zielumgebung erfolgen, da erst hierdurch die ggf. hinterlegten Extension-Scripts geladen und initialisiert werden!

toolbar

```
toolBar: {component}
```

Dieses Objekt stellt über die Eigenschaft *„component“* erleichterten Zugang zu einer ToolBar-Komponente von *Sencha Ext JS* zur Verfügung. Für weitere Informationen siehe deren Dokumentation.

statusBar

```
statusBar: {component: statusBar}
```

Dieses Objekt stellt über die Eigenschaft *„component“* erleichterten Zugang zu einer StatusBar-Komponente von *Sencha Ext JS* zur Verfügung. Für weitere Informationen siehe deren Dokumentation.

toolbar

```
messageBox
```

Dieses Objekt stellt erleichterten Zugang zu einer *MessageBox* von *Sencha Ext JS* zur Verfügung. Für weitere Informationen siehe deren Dokumentation.

8 Methoden

Sofern die Bezeichnungen sprechend sind, erfolgt keine weitere Beschreibung

getRootPath

```
getRootPath: function()
```

Liefert das *„Root“* Verzeichnis der *Web-Applikation*.

getClientIP

```
getClientIP: function()
```

Liefert die *IP-Adresse* des aktuellen *Clients*.

getCallbackUrl

```
getCallbackUrl: function()
```

Dient zum Abrufen einer *„CallBack“ URL* mit deren Hilfe externe Seiten eingebunden werden können.

setCallbackUrl

```
setCallbackUrl: function(func)
```

Parameter	Beschreibung
func	Die zu setzende Callback-Funktion

Setzt die *CallBack-Funktion*, welche ausgeführt wird, sofern die *CallBack URL* aufgerufen wird.

getMaskExtensions

```
getMaskExtensions: function()
```

Liefert das Objekt *Extensions* zurück.

getUser

```
getUser: function()
```

Liefert das *User-Objekt* zurück.

getUserID

```
getUserID: function()
```

Liefert die *User-ID* zurück.

getUserField

```
getUserField: function(field)
```

Parameter	Beschreibung
field	Feldname, dessen Wert zurückgegeben werden soll

Liefert den Wert eines beliebigen Feldes aus der *User-Tabelle* zurück.

getCampaign

```
getCampaign: function()
```

Liefert das *Kampagnen-Objekt* zurück.

getCampaignID

```
getCampaignID: function()
```

Liefert die aktuelle *Kampagnen-ID* zurück.

getCampaignField

```
getUserField: function(field)
```

Parameter	Beschreibung
field	Feldname, dessen Wert zurückgegeben werden soll

Liefert den Wert eines beliebigen Feldes aus der *Campaign-Tabelle* zurück.

getCalltable

```
getCampaignID: function()
```

Liefert die den Datensatz aus der *Calltable* des aktuellen Kunden als Objekt zurück.

getCalltableField

```
getUserField: function(field)
```

Parameter	Beschreibung
field	Feldname, dessen Wert zurückgegeben werden soll

Liefert den Wert eines beliebigen Feldes aus der *Calltable* zurück.

setCalltableField

```
getUserField: function(field)
```

Parameter	Beschreibung
field	Feldname, dessen Wert gesetzt werden soll

Setzt den Wert eines beliebigen Feldes aus der *Calltable*.

ACHTUNG: Die Werte werden erst nach dem Aufruf der terminateCall-Funktion in die Datenbank geschrieben!

getCustom

```
getCustom: function()
```

Liefert das aktuelle *Custom-Objekt* (welches der Aufnahme individueller Werte dient) mit allen verfügbaren Eigenschaften zurück.

getCustomField

```
getCustomField: function(fieldName)
```

Parameter	Beschreibung
fieldName	Feldname

Liefert den Wert eines zuvor mit der Funktion *setCustomField* gespeicherten Feldes.

setCustomField

```
setCustomField: function(fieldName, value)
```

Parameter	Beschreibung
fieldName	Der gewünschte Feldname
value	Der zu speichernde Wert

Setzt den Wert eines benutzerspezifischen Feldes. Ist das Feld nicht vorhanden, so wird dieses angelegt. Benutzerdefinierte Felder bleiben bis zum Aufruf der Funktion *terminate-Call* erhalten und dienen der Übergabe von Werten beim Wechsel zwischen verschiedenen Seiten

getIndicator

```
getIndicator: function()
```

Ruft den Index der aktuellen Rufnummer ab.

setIndicator

```
setIndicator: function(value)
```

Parameter	Beschreibung
value	Der neue Wert für die Eigenschaft (1-9)

Setzt den Index der aktuellen Rufnummer für zukünftige Anrufe.

getCallJobs

```
getCallJobs: function()
```

Liefert alle aktuell vorhandenen Rufaufträge einer Session zurück.

getActiveCallJob

```
getActiveCallJob: function()
```

Liefert den aktuell verwendeten Rufauftrag zurück.

phoneNumberChanged

```
phoneNumberChanged: function(index, phoneNumber)
```

Parameter	Beschreibung
index	Position in der Rufnummernliste
phoneNumber	Neue Rufnummer

Dient zum Ändern einer Rufnummer in der Auflistung der vorhandenen Zielrufnummern. Die Übergabe einer leeren Rufnummer führt zum Löschen aus der Auflistung.

emailAddressChanged

```
emailAddressChanged: function(index, emailAddress)
```

Parameter	Beschreibung
index	Position in der E-Mail-Liste
emailAddress	Neue Adresse

Dient zum Ändern einer E-Mail-Adresse in der Auflistung der vorhandenen E-Mail-Adressen. Die Übergabe einer leeren E-Mail-Adresse führt zum Löschen aus der Auflistung.

addTransferSource

```
addTransferSource: function(name, phoneNumber, dialType, onlyLiveTrans-
fer)
```

Parameter	Beschreibung
Name	Bezeichnung der dritten Partei
phoneNumber	Rufnummer
dialType	1 bedeutet intern (Amt), 2 extern
onlyLiveTransfer	Weiterleitung ist nur möglich, wenn die Verbindung mit der dritten Partei aufgebaut ist.

Mittels dieser Methode kann ein externes (nicht kundenbezogenes) Rufnummernziel hinzufügefügt werden, das der Agent während einer Kundenbearbeitung kontaktieren kann. Es ist dem Agenten auch möglich das Kundengespräch ggf. an dieses weiterzuleiten.

removeTransferSource

```
removeTransferSource: function(phoneNumber)
```

Parameter	Beschreibung
phoneNumber	Rufnummer

Mittels dieser Methode kann über die Rufnummer ein externes (nicht kundenbezogenes) Rufnummernziel entfernt werden.

customerChanged

```
customerChanged: function(key, value)
```

Parameter	Beschreibung
key	Name des zugehörigen Wertes
value	Neuer Wert

Bietet die Möglichkeit der Aktualisierung der angezeigten Kundendaten. Die Übergabe erfolgt als Key/Value Wertepaar. Zulässige Werte für Keys sind *„Firstname“*, *„Lastname“* und *„Company“*.

disableMakeCall

```
disableMakeCall: function(index, state)
```

Parameter	Beschreibung
Index	Position in der Rufnummernliste
State	Der Wert ist ein Boolean: Bei False kann der Anwender diese Rufnummer nicht anwählen.

Mit dieser Funktion kann gesteuert, ob ein Agent eine Rufnummer anwählen darf.

disableSendMessage

```
disableSendMessage: function(index, state)
```

Parameter	Beschreibung
index	Position in der Rufnummernliste
state	Der Wert ist ein Boolean: Bei *False* kann der Anwender keine Nachrichten (z.B. SMS) versenden.

Mit dieser Funktion kann gesteuert werden, ob ein Agent eine Nachricht an diese Rufnummer versenden darf.

disableSendEmail

```
disableSendEmail: function(index, state)
```

Parameter	Beschreibung
index	Position in der E-Mail-Liste
state	Der Wert ist ein Boolean: Bei *False* kann der Anwender keine E-Mail versenden.

Mit dieser Funktion kann gesteuert werden, ob ein Agent eine E-Mail an die angegebene Adresse versenden darf.

newEmailMessage

```
newEmailMessage: function(address, sender)
```

Parameter	Beschreibung
address	Empfänger
sender	Absender

Mittels dieser Methode kann man eine neue E-Mail erstellen.

newSMSMessage

```
newSMSMessage: function(text, phoneNumber, account, background)
```

Parameter	Beschreibung
text	Inhalt der neuen SMS
phoneNumber	Rufnummer des Empfängers
account	Zu verwendendes Benutzerkonto
background	Hintergrund

Mittels dieser Methode kann man eine neue SMS erstellen.

hasEmailDrafts

```
hasEmailDrafts: function()
```

Existieren E-Mail-Entwürfe.

hideOpenedEmails

```
hideOpenedEmails: function()
```

Alle geöffneten E-Mails werden geschlossen.

openEmailDrafts

```
openEmailDrafts: function()
```

Öffnet die E-Mail-Entwürfe.

disableWarningOnDraftClose

```
disableWarningOnDraftClose: function()
```

Deaktiviert den Warnhinweis beim Schließen von E-Mail-Entwürfen.

makeCall

```
makeCall: function(phoneNumber, callerID, local)
```

Parameter	Beschreibung
phoneNumber	Zielrufnummer
callerID	Angezeigte Rufnummer
local	True = interner Anruf

Mit dieser Funktion kann ein externer (nicht kundenspezifischer Anruf) aufgebaut werden.

makeConferenceCall

```
makeCall: function(phoneNumber, callerID, local)
```

Parameter	Beschreibung
phoneNumber	Zielrufnummer
callerID	Angezeigte Rufnummer
local	True = interner Anruf

Mit dieser Funktion kann eine Konferenz aufgebaut werden.

makeCustomerCall

```
makeCustomerCall: function(phoneNumber, callerID)
```

Parameter	Beschreibung
phoneNumber	Zielrufnummer
callerID	Angezeigte Rufnummer

Mit dieser Funktion kann ein kundenspezifischer Anruf aufgebaut werden.

hangupCall

```
hangupCall: function(playFile, callJob)
```

Parameter	Beschreibung
playFile	Optional. Ein Soundfile, der vor dem Auflegen abgespielt werden soll.
callJob	Optional. Der CallJob, der aufgelegt werden soll. Ist kein CallJob angegeben, dann wird der aktive CallJob verwendet.

Mit dieser Funktion wird ein *CallJob* aufgelegt.

transferCall

```
transferCall: function(phoneNumber, callerId, local, callJob)
```

Parameter	Beschreibung
phoneNumber	Optional. Zielrufnummer an den der aktive CallJob transferiert werden soll
callerID	Optional. Angezeigte Rufnummer (nur sofern phoneNumber angegeben)
local	Optional. True = interner Anruf (nur sofern phoneNumber angegeben)
callJob	Optional. Der CallJob, an den der aktive CallJob transferiert werden soll

Mit dieser Funktion wird der aktive *CallJob* entweder zu einer Zielrufnummer oder zu einem anderen (bestehenden) *CallJob* transferiert.

switchCall

```
switchCall: function(callJob)
```

Parameter	Beschreibung
callJob	Optional. Gibt den CallJob an, der aktiv geschaltet werden soll.

Mittels dieser Methode wird der aktive *CallJob* auf *„Hold“* gesetzt und ein anderer *CallJob* aktiviert. Der Parameter kann nur entfallen, sofern zwei aktive *CallJobs* bestehen. Andernfalls wird ein Fehler ausgelöst.

initRecording

```
initRecording: function(code)
```

Parameter	Beschreibung
code	1: Kunde wird initial aufgezeichnet 2: Agent wird initial aufgezeichnet 4: Kundenaufzeichung kann vom Agenten gesteuert warden 8: Agentaufzeichnung kann vom Agenten gesteuert werden 16: Kunde-Button ist nicht sichtbar 32: Agent-Button ist nicht

Mittels dieser Methode können die initialen Bedingungen für eine Aufzeichnung festgelegt werden.

getRecordingState

```
getRecordingState: function()
```

Der *RecordingState* besagt, ob nur der Kunde (1) und/oder nur der Agent (2) aufgezeichnet werden. Der Wert 3 besagt demnach, dass beide Seiten aufgezeichnet werden.

setRecordingState

```
setRecordingState: function(value)
```

Parameter	Beschreibung
value	1: Aufzeichnung des Kunden wird gestartet. 2: Aufzeichnung des Agenten wird gestartet.

Mittels des Setzens des *RecordingState* kann die Aufzeichnung gesteuert werden.

hasRecording

```
hasRecording: function()
```

Besagt, ob eine nicht abgespeicherte Aufzeichnung vorliegt.

saveRecording

```
saveRecording: function(filespec)
```

Parameter	Beschreibung
filespec	Speicherort der Aufzeichnung

Die Aufzeichnung wird beendet und gespeichert.

clearRecording

```
clearRecording: function()
```

Die Aufzeichnung wird ohne Speichern beendet.

createAgentRecording

```
createAgentRecording: function()
```

Mit dieser Funktion kann ein *Recording-Objekt* aus Sicht des Agenten erstellt werden.

Bei dem zurückgelieferten Objekt handelt es sich um ein eigenständiges und gekapseltes *Recording-Objekt*, welches ausschließlich von dem *ACD-Provider* für *ttPhoenix 4* zur Verfügung gestellt wird. Es enthält Eigenschaften für:

RecordingID

```
string RecordingID { get; }
```

Die ID der Aufzeichnung.

Album

```
string Album { get; set; }
```

Das Album der Aufzeichnung enthält normalerweise den Namen oder die ID der Kampagne.

Artist

```
string Artist { get; set; }
```

Der Artist der Aufzeichnung ist normalerweise der Agent.

Comment

```
string Comment { get; set; }
```

Ein Kommentar für die Aufzeichnung

Genre

```
string Genre { get; set; }
```

Das Genre der Aufzeichnung

Title

```
string Title { get; set; }
```

Der Titel der Aufzeichnung

Path

```
string Path { get; set; }
```

Mittels der *Path-Eigenschaft* wird der Speicherort des *Recordings* bestimmt. Dieser Wert kann auch gleich zu Beginn gesetzt (und jederzeit überschrieben) werden, um z.B. bei einem *Socket-Abbruch* dennoch eine Zuordnung zu ermöglichen.

RecordingState

```
RecordingState RecordingState { get; set; }
```

Das *Recording* selbst wird mittels der *RecordingState-Eigenschaft* gesteuert und ggf. implizit gestartet und gestoppt. Diese setzt auch die entsprechenden Kanäle:

Bezeichnung	Wert	Beschreibung
NoRecording	0	keine Aufzeichnung
RecordingIn	1	Der eingehende Kanal wird aufgezeichnet
RecordingOut	2	Der ausgehende Kanal wird aufgezeichnet
RecordingInOut	3	Beide Kanäle werden aufgezeichnet

Hierbei ist zu beachten, dass bei *in/out* jeweils die Sicht bezogen auf den Agenten oder Kunden gilt, also je nach erstelltem Objekt.

SaveRecording

```
void SaveRecording(string path)
```

Parameter	Typ	Beschreibung
path	String	Unter diesem Pfad wird das *Voicefile* abgespeichert.

Mit dieser Funktion wird die Aufzeichnung gespeichert. Nach dem Speichern wird der *RecordingState* automatisch auf 0 (keine Aufzeichnung) gesetzt. Anschließend kann die Aufzeichnung durch das Setzen des *RecordingState* erneut gestartet werden.

clearRecording

```
void ClearRecording()
```

Mit dieser Funktion wird die Aufzeichnung gelöscht. Nach dem Löschen wird der *RecordingState* automatisch auf 0 (keine Aufzeichnung) gesetzt. Anschließend kann die Aufzeichnung durch das Setzen des *RecordingState* erneut gestartet werden.

Dispose

```
void Dispose()
```

Zerstört das Objekt. Wird das Objekt nicht mehr benötigt, so muss dieses zwingend durch den Aufruf der *Dispose-Methode* zerstört werden.

createAgentRecordingEx

```
createAgentRecordingEx: function(type, format)
```

Parameter	Beschreibung
Type	0: keine Angabe 1: Mono 2: Stereo
format	1: PCM 8 Bit 2: PCM 16 Bit 3: A-Law 8 Bit

Diese Methode entspricht der *createAgentRecording*. Es können aber abweichende Parameter für Typ und Format der Aufzeichnung angegeben werden.

createCustomerRecording

```
createCustomerRecording() : Object
```

Mit dieser Funktion kann ein *Recording-Objekt* aus Sicht des Kunden erstellt werden.

Ansonsten entspricht sie der Methode *createAgentRecording*.

createCustomerRecordingEx

```
createCustomerRecording(func)
```

Parameter	Beschreibung
type	1: Mono-Aufzeichnung. Sämtliche Streams werden auf einer Spur gemischt. 2: Stereo-Aufzeichnung. Der ein- und ausgehende Stream werden getrennt auf zwei Spuren aufgezeichnet.
format	1: PCM 8 Bit 2: PCM 16 Bit 3: A-Law 8 Bit

Diese Methode entspricht der *createCustomerRecording*. Es können aber abweichende Parameter für Typ und Format der Aufzeichnung angegeben werden.

initEchoCancellation

```
initEchoCancellation: function(value)
```

Parameter	Beschreibung
value	true: Echokompensation wird aktiviert false: Echokompensation wird deaktiviert.

Mittels dieser Methode kann man die Echo- bzw. Hallunterdrückung des *ACD-Systems* verwenden.

startAgentPlayFile

```
startAgentPlayFile: function(file, state, error)
```

Parameter	Beschreibung
file	Der Pfad des Soundfiles (auf dem Phoenix-Server)
State	Callback-Funktion für des Wechsel des Status
error	Callback-Funktion für Fehler-Meldungen

Bietet die Möglichkeit, dem Agenten einen Soundfile vorzuspielen.

abortAgentPlayFile

```
abortAgentPlayFile: function()
```

Beendet die zuvor gestartete Wiedergabe eines Soundfiles.

startCustomerPlayFile

```
startCustomerPlayFile: function(file, state, error)
```

Parameter	Beschreibung
file	Der Pfad des Soundfiles (auf dem Phoenix-Server)
State	Callback-Funktion für des Wechsel des Status
error	Callback-Funktion für Fehler-Meldungen

Biete die Möglichkeit, dem Kunden ein Soundfile vorzuspielen.

abortCustomerPlayFile

```
abortCustomerPlayFile: function()
```

Beendet die zuvor gestartete Wiedergabe eines Soundfiles.

startDTMFDetection

```
startDTMFDetection: function(timeout, detection, error)
```

Parameter	Beschreibung
timeout	Das Timeout für die Eingabe
State	Callback-Funktion für das Ergebnis der Eingabe
error	Callback-Funktion für Fehler-Meldungen

Bietet die Möglichkeit, eine *DTMF-Detektion* zu starten.

abortDTMFDetection

```
abortDTMFDetection: function()
```

Beendet die zuvor gestartete *DTMF-Detektion*.

requestTransferCallWithMask

```
requestTransferCallWithMask: function(targetCampaignId, targetChannel,
success, failed)
```

Parameter	Beschreibung
targetCampaignId	ID der Ziel-Kampagne
targetChannel	Kommunikationskanal
success	Callback-Funktion für den Erfolgsfall
failed	Callback-Funktion für den Fehlerfall

Mittels dieser Funktion wird ein Agent, der an der Zielkampagne und dem festgelegten Kanal angemeldet ist, zur Weiterbearbeitung angefragt. Sofern ein solcher vorhanden ist, wird automatisch ein Chat-Dialog inkl. der Möglichkeit des Call-Transfers eingeblendet.

ACHTUNG: Innerhalb der Callback-Funktion für den Erfolgsfall (also nach dem Call-Transfer) muss die reguläre TerminateCall-Funktion zum Schließen der Kundenbearbeitungsmaske aufgerufen werden.

showNotification

```
showNotification: function(title, message, delay, callback, scope)
```

Parameter	Beschreibung
title	Überschrift
message	Inhalt der Mitteilung
delay	Anzeigedauer des Fensters (optional)
callback	Callback-Funktion (optional)
scope	Sichtbarkeitsbereich der Funktion (optional)

Bietet die Möglichkeit der Anzeige einer Benachrichtigung.

disableWarningOnDraftClose

```
disableWarningOnDraftClose: function ()
```

Gibt die Möglichkeit den Warnhinweis bei Schließen von E-Mails, ohne vorheriges Senden zu unterbinden.

backToSearchMask

```
backToSearchMask: function ()
```

Bietet die Möglichkeit, zurück zur Kunden-Suchmaske zu gelangen, sofern der Aufruf von dieser erfolgte.

terminateCall

```
terminateCall: function(termCode, appointment, personal, reCall, transferTarget, transferCode, args)
```

Parameter	Beschreibung
termCode	Ein beliebiger Code, der in der Calltable als Gesprächsergebnis hinterlegt werden soll.
appointment	Optional. Gibt das Datum des Rückrufes an, sofern eine Wiedervorlage vereinbart wurde.
personal	Optional. True, sofern die Wiedervorlage dem aktuellen Agenten zugeteilt werden soll.
reCall	Obsolet

transferTarget	Optional. Das Routing, an den der Call zurückverwiesen werden soll
transferCode	Optional. Ein Code, welcher beim Zurückweisen innerhalb des Routings ausgewertet werden kann.

Beendet die Bearbeitung des aktuellen Kunden. Ein eventuell noch bestehendes Gespräch wird automatisch aufgelegt.

Dieser Befehl ist als Abschluss der Bearbeitung zwingend notwendig und schließt die aktuelle Webseite. Sämtliche Variablen werden zurückgesetzt.

searchCustomer

```
searchCustomer(count, options)
```

Parameter	Beschreibung
count	Maximale Anzahl an Datensätzen, die in der Suchmaske angezeigt werden sollen.
options	Auflistung von Schlüssel-Wert-Paaren, die beim Aufruf der Inbound-Prozedur verwendet werden.

Mittels dieser Methode kann man in der Suchmaske Kunden suchen.

addCustomer

```
addCustomer(campaignId)
```

Parameter	Beschreibung
campaingId	ID der Kampagne

Mittels dieser Methode kann man aus der Suchmaske heraus für eine Kampagne einen neuen Kunden hinzufügen.

getCustomer

```
getCustomer(customerId, campaignId)
```

Parameter	Beschreibung
customerId	ID des Kunden
campaignId	ID der Kampagne

Mittels dieser Methode kann man aus der Suchmaske heraus einen vorhandenen Kunden einer bestimmten Kampagne aufrufen.

rejectCustomer

```
rejectCustomer(markedAsJunk)
```

Parameter	Beschreibung
markedAsJunk	Als Spam deklarieren

Mittels dieser Methode kann in der Suchmaske die Bearbeitung eines *WorkItems* abbrechen und den diesen ggf. als Spam deklarieren.

9 Ereignisse

Sofern die Bezeichnungen sprechend sind, erfolgt keine weitere Beschreibung

beforeNewMessage

```
beforeNewMessage: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Bei Rückgabe von *false* wird das Versenden der Nachricht unterbunden.

beforeNewChatMessage

```
beforeNewChatMessage: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Bei Rückgabe von *false* wird das Versenden des Chats unterbunden.

beforeNewEmail

```
beforeNewEmail: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Bei Rückgabe von *false* wird das Versenden der E-Mail unterbunden.

beforeAcceptCall

```
beforeAcceptCall: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Bei Rückgabe von *false* wird das Annehmen des Anrufs unterbunden.

beforeMakeCall

```
beforeMakeCall: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Bei Rückgabe von *false* wird das Initialisieren des Anrufs unterbunden.

callJobAdded

```
callJobAdded: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Dient der Information, dass der Liste der aktuellen Rufaufträge ein Eintrag hinzugefügt wurde.

callJobRemoved

```
callJobRemoved: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Dient der Information, dass aus der Liste der aktuellen Rufaufträge ein Eintrag entfernt wurde.

callJobStateChanged

```
callJobStateChanged: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Dient der Information, dass sich der Status eines Rufauftrags geändert hat.

callJobPlaybackStateChanged

```
callJobPlaybackStateChanged: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Dient der Information, dass sich der Status für das Abspielen einer Ansage eines Rufauftrags geändert hat.

callJobDTMFDetection

```
callJobDTMFDetection: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Dient der Information, dass eine Taste *(DTMF)* bei einem Rufauftrag gedrückt wurde.

callJobMarkedAsWrongNumber

```
callJobMarkedAsWrongNumber: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Kennzeichne, dass ein Rufauftrag als *„falsche Zielrufnummer“* gekennzeichnet wurde.

prepareMessageContent

```
prepareMessageContent: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Gibt die Möglichkeit, vor dem Versenden einer Nachricht diese zuvor anzupassen.

prepareEmailContent

```
prepareEmailContent: {
	addEventHandler: function (fn, scope)
	removeEventHandler: function (fn)
}
```

Gibt die Möglichkeit, vor dem Versenden einer E-Mail diese zuvor anzupassen.

activeCallJobChanged

```
activeCallJobChanged: {
       addEventHandler: function (fn, scope)
       removeEventHandler: function (fn)
}
```

Dient der Information, dass sich der aktuelle Rufauftrag geändert hat.

10 Auflistungen

Sofern die Bezeichnungen sprechend sind, erfolgt keine weitere Beschreibung

CallState

Die Auflistung der Zustände eines Anrufs.

Bezeichnung	Wert	Beschreibung
Disconnected	0	Unverbunden. Der abschließende Status eines Anrufs.
Waiting	1	Der Anruf befindet sich in einer Warteschlange, bis eine Leitung frei wird.
Dialing	2	Die Anwahl wurde initialisiert.
Ringing	3	Klingeln
Analyzing	4	Analyse, ob es sich beim Angerufenen um einen Livespeaker handelt.
Connected	5	Verbunden
Hold	6	Der Anruf befindet sich im Status *„Halten“*
HoldRequest	7	Der Angerufene hat den der Status *„Halten“* angefordert.
HoldReject	8	Die Anforderung des Status *„Halten“* wurde abgelehnt.
Muted	9	Der Anruf ist stumm geschaltet.
Transferring	10	Der Anruf wird zum Agenten transferiert.
TransferReject	11	Die Anforderung eines Transfers vom Angerufenen wurde abgelehnt.
TransferRequest	12	Der Angerufene hat einen Transfer angefordert.
IncomingCall	13	Ein eingehender Anruf wurde erkannt.
ReconnectRequest	14	Ein erneuter Verbindungsaufbau durch den Angerufenen wurde angefordert.
ReconnectReject	15	Die Anforderung eines erneuten Verbindungsaufbaus wurde abgelehnt.
Canceled	96	Der Anruf wird im Dialmode *Preview* beendet ohne dass er angewählt wurde.
Rejected	97	Ein eingehender Anruf wird abgewiesen und zurück in die Warteschleife versetzt.
Assigned	98	Der Anruf ist einem Agenten zugewiesen, wird aber nicht angewählt. Der Dialmode ist *Preview*.
Transferred	99	Ein Anruf wurde von einem Agenten transferiert.

CallDirection

Die Auflistung der möglichen Richtungen des Auftrags.

Bezeichnung	Wert	Beschreibung
Inbound	0	Es handelt sich um eine eingehende Kommunikation.
Outbound	1	Es handelt sich um eine ausgehende Kommunikation.

ContentType

Die Auflistung der möglichen Ausprägungen des Auftrags

Bezeichnung	Wert	Beschreibung
None	0	Der Auftrag wurde ohne initiierende Kommunikation vom Agenten erstellt (Kundensuche)
ACD	1	Der Auftrag wurde aufgrund Anrufs (In- oder Outbound) erstellt.
Chat	2	Der Auftrag wurde aufgrund eines eingehenden Chats erstellt.
Message	4	Der Auftrag wurde aufgrund einer eingehenden Nachricht (z.B. SMS) erstellt.
Email	8	Der Auftrag wurde aufgrund einer eingehenden E-Mail erstellt.

11 Zugriff auf Extensions über die Schnittstelle

Auf *Mask-Extensions* kann über die *„extension“*-Eigenschaft der Schnittstelle direkt zugegriffen werden. Der Zugriff auf eine spezifische Extension kann hierbei über dessen Konfigurations-Key erfolgen.

ACHTUNG: Die Extension-Auflistung muss initial mittels getValue() abgerufen werden. Dieses sollte innerhalb der der Zielumgebung erfolgen, da erst hierdurch die ggf. hinterlegten Extension-Scripts geladen und initialisiert werden!

Beispiel:

```
if (contenInterface.extensions.getValue != null) {
	contenInterface.extensions.getValue();
}

var mySampleExtension = contenInterface.extensions['MySampleExtension'];
```

Zusätzlich kann durch die Extensions via Index iteriert werden:

```
for (var i = 0; i < contenInterface.extensions.length; i++) {
	var extension = contenInterface.extensions[i];
}
```

Auf Extensions von Unterobjekten (z.B. der Kampagne) kann über die jeweilige Extension-Eigenschaft der einzelnen Objekte zugegriffen werden. Aus Performance-Gründen sind diese zumeist ebenfalls *„ByRef“* Implementiert und müssen zuvor vom Server mittels *getValue()* abgerufen werden:

```
var campaignExtensions = contentInterface.Campaign.Extensions.getValue();
```

HINWEIS: Sofern die Implementierung mittels „ByRef“ erfolgte, werden die Client-Scripts dieser Extension erst bei erstmaligem Aufruf der getValue-Funktion geladen und ausgeführt.

12 Automatische Erweiterungen durch Extensions

Eine weitere Option zur Implementierung von Extensions stellt die Möglichkeit der automatischen Ausführung von Scripts durch Extensions dar. Somit ist es z.B. möglich, den Funktionsumfang der Maske zu erweitern, ohne deren Code ändern zu müssen. Hierbei wird ein Skript während der Initialisierung der Schnittstelle ausgeführt:

Beispiel:

```
// define namespace
namespace('ttFrame.Mask.Extension.Sample.Provider.V1');

ttFrame.Mask.Extension.Sample.Provider.V1.Provider = function (remote,
owner) {

    function SampleClientFunction(p) {
        return remote.SampleServerFunction(p);
    }
    owner.MySampleExtension = this;

    return this;

}
```

Der Initialisierungs-Funktion werden standardmäßig die Parameter *„remote“* (entspricht dem *contentInterface.extensions['MySampleExtension']-Objekt*) und das *window-Objekt* der Seite als *„owner“* übergeben.

Das Extension-Skript kann somit nach Belieben auf die Seite zugreifen und Änderungen vornehmen (z.B. sich in das *window-Objekt* schreiben). Über das remote-Objekt kann dann auf die serverseitig freigegebenen Funktionen zurückgegriffen werden.

HINWEIS: Die Ausführung der Initialisierungs-Funktion erfolgt durch den initialize-Befehl der Schnittstelle (s. Abschnitt 6) und vor dem Aufruf der entsprechenden Callback-Funktion.

13 SQL-Prozeduren

13.1 Statistik Prozeduren

SetCallStatistic

```
CREATE PROCEDURE [dbo].[SetCallStatistic]
	@Mandator varchar(255),			-- Mandator
	@User int,				-- User-ID
	@Login varchar(255),			-- User-Login
	@TimeStamp datetime,			-- TimeStamp
	@TermCode varchar(255),			-- TermCode
	@SysCode int,				-- SysCode
	@Direction int,				-- Call-Direction
	@AnalyzingDuration int,			-- Analyzing-Time
	@RingingDuration int,			-- Ringing-Time
	@HoldDuration int,			-- Hold-Time
	@WaitingDuration int,			-- Waiting-Time
	@ReadyDuration int,			-- Ready-Time
	@PredialDuration int,			-- Predial-Time
	@DialingDuration int,			-- Dialing-Time
	@TalkingDuration int,			-- Talking-Time
	@WrapupDuration int,			-- Wrapup-Time
	@TransferDuration int,			-- Transfer-Time
	@ConnectedDuration int,			-- Connected-Time
	@CallID	varchar(255),			-- Call-ID
	@CalledNumber varchar(255),		-- Called Number
	@CallingNumber varchar(255),		-- Calling Number
	@SigninCampaign int,			-- Campaign-ID (Signin)
	@LogicalCampaign int,
	@DataCampaign int,			-- Campaign-ID (Data)
	@Customer int,				-- Customer-ID
	@ForeignKey varchar(255),		-- Foreign-Key
	@BillingCode varchar(255),		-- Billing-Code
	@Cluster varchar(255),			-- Cluster
	@ReferenceCallID varchar(255),		-- ReferenceCall-ID
	@ParentCallID varchar(255),		-- ParentCall-ID
	@SessionID varchar(255),		-- SessionID
	@ExternalExec nvarchar(4000) out	-- External SQL Statement
```

SetCustomStatistic

```
CREATE PROCEDURE [dbo].[SetCustomStatistic]
       @Mandator varchar(255),           -- Mandator
       @User int,                        -- User-ID
       @Login varchar(255),              -- User-Login
       @TimeStamp datetime,              -- TimeStamp
       @CustomCode varchar(255),         -- Code
       @CustomValue float,               -- Value
       @Direction int,                   -- Call-Direction
       @CallID      varchar(255),        -- Call-ID
       @CalledNumber varchar(255),       -- Called Number
       @CallingNumber varchar(255),      -- Calling Number
       @SigninCampaign int,              -- Campaign-ID (Signin)
       @DataCampaign int,                -- Campaign-ID (Data)
       @Customer int,                    -- Customer-ID
       @ForeignKey varchar(255),         -- Foreign-Key
       @BillingCode varchar(255),        -- Billing-Code
       @Cluster varchar(255),            -- Cluster
       @ReferenceCallID varchar(255),    -- ReferenceCall-ID
       @ParentCallID varchar(255),       -- ParentCall-ID
       @SessionID varchar(255),          -- SessionID
       @ExternalExec nvarchar(4000) out  -- External SQL Statement
```

SetUserStatistic

```
CREATE PROCEDURE [dbo].[SetUserStatistic]
       @Mandator varchar(255),           -- Mandator
       @User int,                        -- User-ID
       @Login varchar(255),              -- User-Login
       @State int,                       -- State
       @TimeStamp datetime,              -- TimeStamp
       @Duration int,                    -- Duration
       @Direction int,                   -- Call-Direction
       @SigninCampaign int,              -- Campaign-ID (Signin)
       @LogicalCampaign int,
       @DataCampaign int,                -- Campaign-ID (Data)
       @BillingCode varchar(255),        -- Billing-Code
       @Cluster varchar(255),            -- Cluster
       @ExternalExec nvarchar(4000) out  -- External SQL Statement

    -- 0 Unknown
       Unknown state (might be useful for the consuming application)
    -- 1 Ready
       The Agent is logged on and is ready to receive calls.
    -- 2 PreReady
       The Agent is logged on and is not yet ready to receive calls,
       but the dialer can begin to dial out for the agent.
    -- 3 NotReady
       The Agent is logged on and is not ready to receive calls.
    -- 4 PostProcessing
       The Agent is logged on and is currently doing the
       post processing work.
    -- 5 PreProcessing
       The Agent is logged on and is not ready and has open
       the customer dialog.
    -- 6 Talking
       The Agent has a call.
    -- 7 Dialing
       The Agent is dialing.
    -- 8 ExterNalTaking
       The Agent is External Talking.
    -- 100
       Pause (default)
    -- 101-199
       Pause (custom)
```

13.2 Kampagnen Prozeduren

Appointments

CP_000xxx_Appointment =>: xxx = Kampagnen ID

```
CREATE PROCEDURE [dbo].[CP_000xxx_Appointments]
	@Top int,
	@Owner int = 000xxx,
	@Mode int = 2,
	@Filter int = 24

	-- 1   Normal callJobs
	-- 2   General appointments
	-- 4   Released personal appointments
	-- 8   Prio calls (Lost calls)
	-- 16  Personal appointments
	-- 64  Enable user check for personal appointments
	-- 128 Enable timeframe restriction
	-- 256 Disable filter
	-- 512 OverDial (+1)
	-- 1024 Disable counter restrictions
```

Finalize

CP_000xxx_Finalize =>: xxx = Kampagnen ID

```
CREATE PROCEDURE [dbo].[CP_000xxx_Finalize]
	@Campaign int,
	-- owner-campaign id
	@Customer int,
	-- customer id (Internal_ID)
	@ForeignKey varchar(255),
	-- foreign customer key
	@Resultcode int,
	-- dialer call result (see result code enumeration)
	@Termcode varchar(255),
	-- user defined call result (only when connected)
	@CallDirection int,
	-- call direction (see call direction enumeration)
	@State int,
	-- current customer state (see enumeration)
	@Indicator int,
	-- current phone indicator
	@TargetAddress varchar(255),
	-- current target phone number
```

```
@CallerID varchar(255),
-- current calling phone number
@LastAccess datetime,
-- last access
@LastAttempt datetime,
-- last attempt
@User int,
-- user id (Agent)
@Count int,
-- cuurent count
@DropCount int,
-- current drop count
@LostCount int,
-- current lost count
@DayCount int,
-- current count per day
@WeekCount int,
-- current count per week
@MonthCount int,
-- current count per month
@AppointmentCount int,
-- current count of appointment
@MaxCount int,
-- campaign setting max count
@MaxDropCount int,
-- campaign setting max drop count
@MaxLostCount int,
-- campaign setting max lost count
@MaxDayCount int,
-- campaign setting max count per day
@MaxWeekCount int,
-- campaign setting max count per week
@MaxMonthCount int,
-- campaign setting max count per month
@MaxAppointmentCount int,
-- campaign setting max count of general appointment
@MaxPeronalAppointmentCount int,
-- campaign setting max count of personal appointment
@NewState int,
-- new customer state (see enumeration)
@NewIndicator int,
-- new phone indicator
@NewAppointment datetime,
-- new appointment date
@NewAppointmentUser int,
-- new appointment user (0 = all users)
@NewMinDate datetime,
-- new min date for the next call
```

```
@NewMaxDate datetime,
-- new max date for the next call
@NewMinTime datetime,
-- new min time for the next call
@NewMaxTime datetime,
-- new max time for the next call
@NewCount int,
-- new count
@NewDropCount int,
-- new drop count
@NewLostCount int,
-- new lost count
@NewDayCount int,
-- new count per day
@NewWeekCount int,
-- new count per week
@NewMonthCount int,
-- new count per month
@NewAppointmentCount int,
-- new count of appointment
@IsCounted int,
-- indicate, that the Call is counted
@NewDialmode int,
-- Dialmode for future calls
-- (otional, 0 = default: 1 = predictive, 2 = ringing, 3 = preview)
@NewDropMode int,
-- Dropmode for future calls
-- (otional, 0 = default: 1 = not droppable, 2 = droppable)
@NewTimeFrameRestriction int,
-- TimeFrameRestriction for future calls
-- (otional: 0 = default, 1 = on, 2 = off)
@NewCallerID varchar(255)
-- CallerID for future calls (otional)
```

Inbound

CP_000xxx_Inbound =>: xxx = Kampagnen ID

```
CREATE PROCEDURE [dbo].[CP_000xxx_Inbound]
	@Top int,
	@Phone varchar(255) = '',
	@Firstname varchar(255) = '',
	@Lastname varchar(255) = '',
	@Factory varchar(255) = ''
```

Outbound

CP_000xxx_Outbound =>: xxx = Kampagnen ID

```
CREATE PROCEDURE [dbo].[CP_000xxx_Outbound]
       @Top int ,
       @Owner int = 000xxx,
       @Mode int = 2,
       @Filter int = 15
```

Lock

CP_000xxx_Lock =>: xxx = Kampagnen ID

```
CREATE PROCEDURE [dbo].[CP_000xxx_Lock]
       @ID int = 0,
       @Owner int = 000xxx,
       @Mode int = 0
```

Unlock

CP_000xxx_Unlock =>: xxx = Kampagnen ID

```
CREATE PROCEDURE [dbo].[CP_000xxx_Unlock]
       @ID int = 0,
       @MOwner int = 000xxx,
       @Mode int = 0
```

14 Datenbankstruktur

14.1 Kampagnenspezifische Objekte

Das ttFrame greift auf genormte Objekte zu, die in Form von Tabellen, Sichten und Prozeduren auf der Datenbank abliegen.

Übersicht

Kurzbeschreibung aller Datenbank Objekte einer Basis-Kampagne. In diesem Fall werden folgende Objekte angelegt.

HINWEIS: Bitte beachten Sie, dass bei anderen Kampagnen-Typen ggf. manche Tabellen als Sichten erstellt werden, die wiederum auf Datenbankobjekte der Referenz-Kampagne verweisen.

Bezeichnung	Objekttyp	Beschreibung
Calltable	Tabelle	Die „*Calltables*" enthalten die Datensätze, die im Rahmen der Kampagne bearbeitet werden sollen. Diese Tabellen können Sie beliebig um weitere Spalten erweitern. *ACHTUNG: Bei dieser Tabelle handelt es sich um eine Kundenspezifische Tabelle! Die mitgelieferte Tabelle dient nur der Anschauung und als Beispiel.*
Locks	Tabelle	Die *„Locks"* beinhaltet die bereits aufgerufenen Datensätze. Hiermit wird verhindert, dass Datensätze doppelt aufgerufen werden.
Internal	Sicht	Ist die zentrale Sicht, welche die Daten für die Bearbeitung vorhält. In dieser Sicht werden die Daten in eine genormte Struktur gebracht, damit der Kampagnen-Provider damit arbeiten kann. Alle Namen dieser Felder beginnen mit *Internal_*. Sofern es sich nicht um ein tatsächliches Feld der zugrundeliegenden Tabelle handelt, muss es wenigstens als *Alias-Feld* in der Sicht existieren. Als Sicht definiert, können die Daten einer bereits vorhandenen Tabelle auf die benötigte Struktur zugeordnet werden.
Filter	Sicht	Dient der Einschränkung der Datensätze zum Abruf durch den *Dialer*. Hier werden Filterkriterien definiert die generell für diese Kampagne gelten und nicht über den *Customfilter* anpassbar sein sollen.

Customfilter	Sicht	Dient der individuellen Filterung der Datensätze über die Oberfläche.
Outbound	Prozedur	Dient dem Abrufen von Datensätzen für ausgehende Anrufe.
Inbound	Prozedur	Dient dem Abruf (Suche) von Datensätzen bei eingehenden Anrufen.
Appointment	Prozedur	Dient dem intervallmäßigen Abrufen von Wiedervorlagen, sofern kein Benutzerkalender genutzt wird.
Finalize	Prozedur	Rückgabe und Speicherung der über das *ttFrame* erzielten Ergebnisse und Wählparameter aus der Kampagne.
Unlock	Prozedur	Freigabe abgerufener Datensätze nach Beendigung einer Kampagne.

Die Syntax dieser Objekte lautet wie folgt:

`CP_XXXXXX_Bezeichnung`

XXXXXX steht hierbei für die interne Kampagnen-ID.

Beschreibung der Filterstruktur

Die ersten drei Sichten, welche im Folgenden beschrieben werden, dienen zum Aufbau der Filterstruktur im Hintergrund:

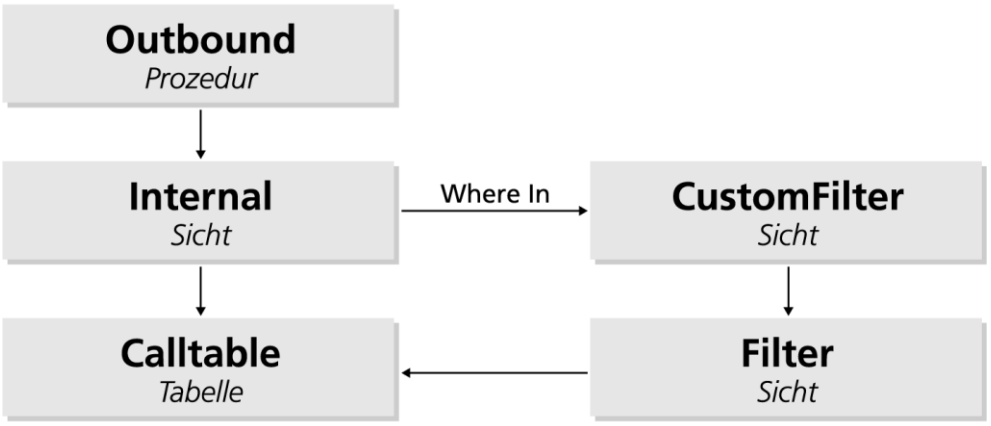
Es besteht die Möglichkeit im *ttFrame* eine eigene *Calltable* oder die *ttFrame-Calltables* zu verwenden. *Calltables* enthalten die Datensätze, die im Rahmen der Kampagne bearbeitet werden sollen.

Dreh- und Angelpunkt der Filterstruktur und der Auswahl der Datensätze für die Kampagne ist die Sicht `CP_XXXXXX_Internal`.

Die Sicht `CP_XXXXXX_Internal` kann durch weitere Felder aus eigenen *Calltables* ergänzt werden. Die Standardfelder *[Internal_]* werden in den folgenden Abschnitten `CP_XXXXXX_Calltable` bzw. `CP_XXXXXX_Internal` mit ihren Bedeutungen beschrieben.

Die Filter, welche über die Oberfläche im Bereich Filter definiert werden, sind in der Sicht `CP_XXXXXX_CustomFilter` hinterlegt. Diese Sicht wird beim Abspeichern der eingestellten Filterkriterien erzeugt. In der Sicht `CP_XXXXXX_CustomFilter` werden die Datensätze aus der Sicht `CP_XXXXXX_Filter` weiter eingeschränkt.

Die Sicht `CP_XXXXXX_Filter` wiederum enthält alle Datensätze aus der Tabelle *Calltable*. In dieser Sicht können darüber hinaus weitere Einschränkungen definiert werden. Die generell für diese Kampagne gelten und nicht über den *Customfilter* anpassbar sein sollen. Eine Anpassung der Sicht auf die Begebenheiten der *Calltable* ist erforderlich.



stark vereinfachte exemplarische Darstellung des Datenzugriff via Outbound

14.2 Objekte der Datenbank

Tabelle: *CP_XXXXXX_BlackList*

Die *„BlackList“* beinhaltet alle Fremdschlüssel von Kunden, die nicht angewählt werden dürfen.

```
CREATE TABLE [dbo].[CP_######_BlackList]

[ForeignKey] [varchar](50)
-- Fremdschlüssel
```

Tabelle: *CP_XXXXXX_Locks*

Die *„Locks“* beinhaltet die bereits aufgerufenen Datensätze. Hiermit wird verhindert, dass Datensätze doppelt aufgerufen werden.

```
CREATE TABLE [dbo].[CP_######_Locks]

[ID] [int]
-- ID des Locks
[Owner] [int]
-- ID der Kampagnen
[Mode] [int]
-- Mode
[Reference]
-- Interne Referenz für den Datenabgleich beim Aufrufen der Datensätze
[LockDate] [datetime]
-- Datum des Eintrags
```

Sicht: *CP_XXXXXX_Internal*

Ist die zentrale Sicht, welche die Daten zur Bearbeitung vorhält. In dieser Sicht werden die Daten in eine genormte Struktur gebracht, damit der Kampagnen-Provider damit arbeiten kann.

Die folgende Feldauflistung beschreibt die Mustersicht, welche beim Hinzufügen einer Kampagne angelegt wird.

Bezeichnung	Beschreibung
Internal_ID	Kunden-ID
Internal_State	Status des Kunden
Internal_AccessCount	Anzahl Aufrufe des Kunden
Internal_LastAccess	Letzter Aufruf des Kunden
Internal_LastCode	Letzter Term-Code des Kunden
Internal_LastSysCode	Letzter System-Code des Kunden
Internal_LastAttempt	Letzter Aufruf des Kunden
Internal_LastUser	Letzter Benutzer
Internal_Appointment	Termin der Wiedervorlage
Internal_AppointmentUser	Benutzer einer persönlichen Wiedervorlage
Internal_AppointmentCount	Anzahl an Wiedervorlagen für diesen Kunden
Internal_Filter	Frei definierbarer Filter
Internal_Dropable	Darf ein Verbindungsaufbau vom ACD-System abgebrochen werden?
Internal_DropCount	Anzahl Drops für diesen Kunden
Internal_LostCount	Anzahl Lost für diesen Kunden
Internal_Dialmode	Wählverhalten des Datensatzes
Internal_DropMode	Dropmodus - Kennzeichen für "Agent reserviert" auf Datensatzebene
Internal_TimeFrameRestriction	Prüfung der Zeitrahmenbeschränkung
Internal_CallerID	Zu übertragende Rufnummer für den Datensatz
Internal_MinDate	Frühestes Aufrufe-Datum (Startdatum)
Internal_MaxDate	Letztes Aufrufe-Datum (Stoppdatum)
Internal_MinTime	Früheste Aufrufe Uhrzeit (Zeitfenster – Beginn)
Internal_MaxTime	Späteste Aufrufe Uhrzeit (Zeitfenster – Ende)
Internal_BlockedUntil	Gesperrt bis
Internal_Count	Anzahl Anwahlversuche
Internal_DayCount	Anzahl Anwahlversuche pro Tag
Internal_WeekCount	Anzahl Anwahlversuche pro Woche
Internal_MonthCount	Anzahl Anwahlversuche pro Monat
CUSTOMERID	Fremdschlüssel zum Kunden
INDICATOR	Indikator für die aktuelle Rufnummer

HOME	Rufnummer
BUSINESS	Rufnummer
OTHER	Rufnummer
SALUTATION	Anrede
TITLE	Titel
FIRSTNAME	Vorname
LASTNAME	Nachname
COMPANY	Firma
STREET	Straße
NUMBER	Hausnummer
ZIP	Postleitzahl
CITY	Stadt

14.3 Allgemeine Objekte des ttFrame

Im Folgenden werden die Standard-Tabellen des *ttFrame* erläutert. Die beschriebenen Tabellen enthalten u.a. Pausengründe, erfasste Zeiten und Werte aus dem *ttFrame*.

Es ist zu beachten, dass jederzeit eigene Tabellen auf der Datenbank hinzugefügt werden können bzw. dass durch das Hinzufügen von neuen Providern neue Tabellen entstehen können.

Tabelle: *Breaks*

In der Tabelle `Breaks` können Sie Ihre eigenen Pausengründe hinterlegen. Neu definierte Pausengründe müssen parallel in der Tabelle `TimeCodes` eingetragen werden.

Bezeichnung	Beschreibung
Break_ID	Interne ID der Pause
Break_Name	Bezeichnung der Pause in der Oberfläche
Break_Order	Die Pausengründe werden gemäß diesem Feld sortiert.

Tabelle: *CallStatistic*

Die Tabelle `dbo.CallStatistic` enthält alle statistischen Ergebnisse zu den einzelnen Anrufen, welche über *ttFrame 4* durchgeführt wurden. Dazu zählen u.a. über welchen Mandanten die Anmeldung erfolgte, wann wurde der Anruf durchgeführt und wie lange wurde telefoniert.

Bezeichnung	Beschreibung
CallStatistic_Index	laufende Nummerierung (Auto Wert) der getätigten Anrufe
CallStatistic_Mandator	Mandant, über welchen der Anruf getätigt wurde
CallStatistic_User	Benutzerbezeichnung (interne User ID) des Users, welcher den Anruf bearbeitet hat
CallStatistic_Login	Benutzername (Login-Name) des Users, welcher den Anruf bearbeitet hat
CallStatistic_TimeStamp	Zeitstempel, wann der Datensatz in die Statistik eingetragen wurde.
CallStatistic_TermCode	Übergebener Term-Code, Festzulegen in der *Finalize* -Funktion
CallStatistic_SysCode	System-Code für das Ergebnis, mit welchem der Anruf beendet wurde
CallStatistic_CustomCode	Der zu setzende Wert kann über die Prozedur angepasst werden. Parallel muss immer der Wert `CallStatistic_CustomValue` gesetzt werden.
CallStatistic_Direction	gibt die Richtung an, in welche der Anruf geführt wurde 1 = eingehend (Inbound) 2 = ausgehend (Outbound)
CallStatisitic_WaitingTime	gibt die Zeit in Millisekunden an, bevor der Wahlvorgang gestartet wurde.
CallStatistic_ReadyTime	gibt die Zeit in Millisekunden an, welche der Agent in Bereitschaft war.
CallStatistic_PredialTime	gibt die Zeit in Millisekunden an, welche der Agent in Vorbereitung war.
CallStatistic_DialTime	gibt die Zeit in Millisekunden an, welche für die Anwahl des Anrufs benötigt wurde.
CallStatistic_TalkTime	gibt die Zeit in Millisekunden an, welche für das eigentliche Gespräch benötigt wurde.

CallStatistic_WrapupTime	gibt die Zeit in Millisekunden an, welche der Agent für die Nacharbeit benötigte.
CallStatistic_CustomValue	Der zu setzende Wert kann über die Prozedur angepasst werden. Parallel muss immer der Wert `CallStatistic_CustomValue` gesetzt werden.
CallStatistic_CallID	automatisch erzeugte eindeutige ID des Anrufs
CallStatistic_CalledNumber	Rufnummer, zu welcher der Anruf aufgebaut wurde
CallStatistic_CallingNumber	Rufnummer, über welche der Anruf aufgebaut wurde (Absenderrufnummer)
CallStatistic_SigninCampaign	Kampagnen-ID der Kampagne, über welche der Benutzer angemeldet ist
CallStatistic_DataCampaign	Kampagnen-ID der Kampagne, aus welcher die Kundendaten stammen (Datenherkunft)
CallStatistic_LogicalCampaign	Kampagnen-ID der Kampagne, welche die Zuteilung der Kundendatensätze übernimmt
CallStatistic_Customer	interne Kunden-ID aus der Calltable
CallStatistic_ForeignKey	interner Fremdschlüssel aus der Calltable
CallStatistic_ReferenceCallID	Die ID des Anrufs, die im Falle eines Transfers beteiligt war.
CallStatistic_ParentCallID	Die ID des Anrufs aufgrund dessen eine Session initiiert wurde
CallStatistic_SessionID	ID der Session

Auflistung: *Callstatistic_SysCodes*

Die *SysCodes* innerhalb von *ttFrame 4* enthalten vordefinierte Ergebniswerte, mit welchen ein Anruf abgeschlossen werden kann.

Bezeichnung	SysCode	Beschreibung
Unknown	0	Unbekanntes Ergebnis
Connected	1	der Anruf wurde erfolgreich an einen Agenten durchgestellt
HardwareError	2	ein Hardwarefehler trat auf
GeneralError	3	ein nicht näher spezifizierter Fehler trat auf

Blacklisted	4	es existiert ein *Blacklist-Eintrag* für die anzuwählende Rufnummer
Dropped	5	der Anrufversuch wurde vom Dialer vor dem Verbindungsaufbau abgebrochen *(Drop)*
Lost	6	der Anruf wurde nach einem Verbindungsaufbau entweder vom Angerufenen oder vom *Dialer* aufgelegt, bevor er zu dem Agenten durchgestellt werden konnte.
NoLiveSpeaker	7	es wurde keine natürliche Person *(Livespeaker)* erkannt
NoAnswer	8	keine Annahme und die maximale Klingelzeit wurde erreicht
Busy	9	Besetzt
InvalidNumber	10	Ungültige Rufnummer
Rejected	11	der Anruf wurde vom Angerufenen abgelehnt
Forwarded	12	der Anruf wurde weitergeleitet
InvalidTrunkGroup	13	die *Trunkgroup* konnte nicht gefunden werden
NotDialed	14	der Anruf wurde von einem Agenten während des Zustands *„Pre Assigned“* nicht angewählt
Abandoned	15	ein eingehender Anruf wurde abgebrochen, bevor an einen Agenten durchgestellt werden konnte
InboundRejected	16	ein eingehender Anruf wurde aufgelegt, weil … a) die Kapazität der Warteschlange nicht ausreichend war b) die zugeordnete Kampagne ungültig war c) keine Kampagne zugeordnet war
InboundDropped	17	ein eingehender Anruf wurde aufgelegt, weil die maximale Wartezeit überschritten wurde
AgentNotFound	18	ein zugeordneter Agent konnte nicht gefunden werden
AddressRejected	19	die angezeigte Rufnummer war ungültig
LeftOnGreeting	20	ein eingehender Anruf wurde während des Abspielens der Willkommensnachricht aufgelegt
VoiceMail	22	die angerufene Partei spielte ein *Voicefile* ab
InboundQuit	23	ein eingehender Anruf wurde aufgelegt ohne, dass er in eine Warteschlange gelangte.

InboundTransferred	24	ein eingehender Anruf wurde transferiert
Deleted	25	der Anruf wurde aus einer Warteschlange gelöscht
NotTransferedToAgent	26	der Anruf wurde zu keinem Agenten durchgestellt
LatestDialingStartTime-Reached	27	die Anwahl wurde nicht durchgeführt, da die aktuelle Anwahlzeit erreicht wurde
InactiveInboundTarget	28	der Inbound-Anruf wurde aufgelegt oder abgebrochen, weil das Ziel nicht aktiv war
NoChannels	29	stand keine Kanal für die Anwahl zur Verfügung
DialingTimeout	30	die maximale Anwahlzeit wurde überschritten
Fax	50	es wurde ein Faxgerät erkannt
NotAvailable	51	nicht verfügbar
VoiceMailBecause-InboundRejected	55	eine eingehende VoiceMail wurde aufgelegt, weil … a) die Kapazität der Warteschlange nicht ausreichend war b) die zugeordnete Kampagne ungültig war c) keine Kampagne zugeordnet war
VoiceMailBecause-InboundDropped	56	eine eingehende VoiceMail wurde aufgelegt, weil die Wartezeit überschritten wurde
VoiceMailBecause-InboundQueueEmpty	57	Inbound-Warteschlange ist leer
VoiceMailBecause-InboundOutOfBusiness-Hours	58	außerhalb der Geschäftszeiten
ChatRejected	83	ein eingehender Chat wurde abgewiesen
SMSRejected	84	eine eingehende SMS wurde abgewiesen
EmailRejected	85	eine eingehende E-Mail wurde abgewiesen
Administration	86	Verwaltungsmodus *(s. ContentType None)*
Chat	87	ein eingehender Chat wurde bearbeitet
SMS	88	eine eingehende SMS wurde bearbeitet
Email	89	eine eingehende E-Mail wurde bearbeitet
Updated	99	aktualisiert

Tabelle: *SystemCodes*

Die Tabelle dient ausschließlich zur Verlinkung mit der Statistiktabelle. Programmtechnisch wird diese nicht verwendet. Es besteht die Möglichkeit pro Mandanten mit individuellen Bezeichnungen für System-Codes zu arbeiten.

Bezeichnung	Beschreibung
SystemCode_ID	interne ID
SystemCode_Mandator	Mandant
SystemCode_Name	individuelle Bezeichnung für einen System-Code
SystemCode_Order	Sortierreihenfolge

Tabelle: *TimeCodes*

Die Tabelle dient ausschließlich zur Verlinkung mit der Statistiktabelle. Programmtechnisch wird diese nicht verwendet. Es besteht die Möglichkeit pro Mandanten mit individuellen Bezeichnungen für *UserStatistic-States* zu arbeiten.

Bezeichnung	Beschreibung
TimeCode_ID	interne ID
TimeCode_Mandator	Mandant
TimeCode_Name	individuelle Bezeichnung für einen *UserStatistic-State*
TimeCode_Order	Sortierreihenfolge

Tabelle: *UserStatistic*

Die Tabelle `dbo.UserStatistic` enthält alle statistischen Ergebnisse zu den einzelnen Benutzern, welche an *ttFrame 4* angemeldet waren. Dazu zählen z.B. über welchen Mandanten die Anmeldung erfolgte, wann erfolgte die Anmeldung und wie lange war der Benutzer angemeldet.

Bezeichnung	Beschreibung
UserStatistic_ID	laufende Nummerierung *(Auto Wert)* der getätigten Anmeldungen

UserStatistic_Mandator	Mandant, über welchen die Anmeldung erfolgte
UserStatistic_User	Benutzerbezeichnung *(interne User ID)* des Users, über welchen die Anmeldung erfolgte
UserStatistic_Login	Benutzername *(Login-Name)* des Users, über welchen die Anmeldung erfolgte
UserStatistic_State	*s. Tabelle TimeCode – TimeCodeID*
UserStatistic_TimeStamp	Zeitpunkt (Datum und Uhrzeit), ab wann sich der Benutzer in dem unter `[UserStatistic_State]` erfassten Status befand
UserStatistic_Duration	Dauer, in welcher sich der Agent in dem unter `[User-Statistic_State]` hinterlegten Status befand
UserStatistic_Direction	gibt den Bereich an, in welchen der Benutzer angemeldet war 1 = eingehend (Inbound) 2 = ausgehend (Outbound)
UserStatistic_SigninCampaign	Kampagnen-ID der Kampagne, über welche der Benutzer angemeldet ist
UserStatistic_DataCampaign	Kampagnen-ID der Kampagne, aus welcher die Kundendaten stammen (Datenherkunft)
UserStatistic_LogicalCampaign	Kampagnen-ID der Kampagne, welche die Zuteilung der Kundendatensätze übernimmt
UserStatistic_BillingCode	Der in der Kampagne festgelegte Abrechnungscode wird hier hinterlegt.
UserStatistic_Cluster	Der in der Kampagne festgelegte Daten Cluster wird hier hinterlegt.

Auflistung: *Userstatistic_State*

Die Status innerhalb von *ttFrame 4* enthalten vordefiniert Werte, mit welchen der Zustand eines Benutzers angegeben wird. Die Status innerhalb von *ttFrame 4* enthalten vordefiniert Werte, mit welchen der Zustand eines Benutzers angegeben wird.

Bezeichnung	State	Beschreibung
Unknown	0	der Status des Agenten ist unbekannt
Ready	1	der Agent ist an der Telefonie angemeldet und bereit für die Annahme von Anrufen

PreReady	2	der Agent ist an der Telefonie angemeldet, aber noch nicht bereit für die Annahme von Anrufen. Das *ACD-System* kann aber bereits Anrufe für den Agenten generieren.
NotReady	3	der Agent ist an der Telefonie angemeldet, aber nicht bereit für die Annahme von Anrufen
PostProcessing	4	der Agent ist an der Telefonie angemeldet und befindet sich in der Nachbearbeitungszeit
PreProcessing	5	der Agent ist an der Telefonie angemeldet und befindet sich in der Kundenbearbeitungsmaske. Er ist nicht bereit für die Annahme von Anrufen.
Talking	6	der Agent befindet sich in einem Gespräch
Dialing	7	der Agent befindet sich in der Anwahlphase eines Anrufs
ExternalTalking	8	der Agent befindet sich in einem externen Gespräch
Unknown	11	der Agent befindet sich in einem unbestimmten Status
Pause	100	Pause *(default)*
Pause	101 - 199	Pause *(custom)* - Benutzer definierte Pausenwerte, welche innerhalb von *ttFrame* hinterlegt werden können

Tabelle: *SessionStatistic*

Die Tabelle `dbo.SessionStatistic` enthält alle statistischen Ergebnisse zu den einzelnen Sessions, welche in *ttFrame 4* gestartet wurden. Dazu zählen z.B. der Anfang und das Ende einer Session, der Typ einer Session u.a.

Ein Tab/Register ist in *ttFrame 4* mit dem *Parallel Processing* eine Session. Hat ein Agent drei Tabs/Register (z.B. zwei Kunden und die Suchmaske) offen, dann stellen diese drei Sessions dar.

Eine Session ist aktiv, solange der Agent den Tab/Register im Vordergrund hat und bearbeitet. Wechselt der Agent auf einen anderen Tab, wechselt er die Session. Die vorher bearbeitete Session ist dann inaktiv.

Bezeichnung	**Beschreibung**
SessionStatistic_Index	Auto Wert (interne ID)
SessionStatistic_SessionId	ID der Session
SessionStatistic_Begin	Beginn der Session

SessionStatistic_End	Ende der Session
SessionStatistic_ActiveDuration	Dauer in Millisekunden in der das Tab für die Bearbeitung der Session ausgewählt war.
SessionStatistic_InactiveDuration	Dauer in Millisekunden in der ein anderer Tab (z.B. andere Session) ausgewählt war.
SessionStatistic_User	Interne User-ID
SessionStatistic_Login	Login Name
SessionStatistic_ContentType	Anlass der Bearbeitung (z.B. Anruf, E-Mail, Chat) None = 0 ACD = 1 Chat = 2 Message = 4 Email = 8
SessionStatistic_TermCode	Übergebener Term-Code, Festzulegen in der *Finalize* Funktion
SessionStatistic_SigninCampaign	Angemeldete Kampagne
SessionStatistic_LogicalCampaign	Kampagne für die Zuteilung des Kunden
SessionStatistic_DataCampaign	Kampagne der Datenherkunft
SessionStatistic_Customer	Interne Kunden-ID aus der Calltable
SessionStatistic_ForeignKey	Möglicher Fremdschlüssel des Kunden
SessionStatistic_Direction	Anrufrichtung (In- oder Outbound) Inbound = 0 Outbound = 1

Tabelle: *WorkItemStatistic*

Die Tabelle `dbo.WorkItemStatistic` enthält alle statistischen Ergebnisse zu den einzelnen *WorkItems*, welche in *ttFrame 4* erzeugt wurden. Dazu zählen z.B. die Art des *WorkItems*, der Zeitpunkt der Zuteilung u.a.

Bezeichnung	**Beschreibung**
WorkItemStatistic_Id	automatisch vergebene interne ID
WorkItemStatistic_SessionId	interne ID der Session

WorkItemStatistic_Type	Art des *WorkItems* (auch *ContentType* genannt) None = 0 ACD = 1 Chat = 2 Message = 4 Email = 8
WorkItemStatistic_SourceKey	Schlüssel der Source
WorkItemStatistic_SourceName	Bezeichnung der Source
WorkItemStatistic_ServiceLevelStartTi-meStamp	Zu diesem Zeitpunkt ist das *WorkItem* erstmals in eine Warteschlange einsortiert worden.
WorkItemStatistic_Assigned-TimeStamp	Zeitpunkt der Zuteilung an einem Agenten
WorkItemStatistic_IsInServiceLevel	Wurde der Servicelevel eingehalten
WorkItemStatistic_IsDirectTransfered	Wurde das *WorkItem* direkt weitergeleitet
WorkItemStatistic_Reference	Absenderkennung (z.B. E-Mail-Adresse des Absenders)

Prozedur: `SetCallStatistic`

Die statistischen Kennzahlen des Anrufs werden mittels dieser Prozedur in die Tabelle `CallStatistic` abgespeichert.

Prozedur: `SetCustomStatistic`

Die Prozedur `SetCustomStatistic` dient zum Setzen individueller statistischer Werte, welche ebenfalls in der Tabelle `CallStatistic` abgespeichert werden.

Prozedur: `SetUserStatistic`

Die statistischen Kennzahlen des Agenten werden mittels dieser Prozedur in die Tabelle `UserStatistic` abgespeichert.

Prozedur: `SetSessionStatistic`

Die statistischen Kennzahlen der Session werden mittels dieser Prozedur in die Tabelle `CallStatistic` abgespeichert.

15 Impressum

Herausgeber

tribe technologies GmbH
Lurgiallee 12
60439, Frankfurt am Main

Sitz und Amtsgericht: Frankfurt am Main, Hrb 47890
Geschäftsführer: Dirk Bunar, Axel Reddehase

Erscheinungsweise: mit neuen Produkt-Versionen
Erscheinungsjahr: 2021
Erscheinungsort: Frankfurt am Main

Fragen und Anregungen zu diesem Dokument senden Sie bitte an unseren Vertriebspartner.

ttUnited GmbH
Lurgiallee 12
60439, Frankfurt am Main

Telefon: +49 69 770 699 - 299
E-Mail: support@ttunited.com