SERVICEWARE

Administrationshandbuch

Teil 8 – Server Scripting

SERVICEWARE

Impressum

helpLine CLM AG

Haldenstrasse 5

6340 Baar

Verwaltungsrat:

Dr. Peter Nicolai Damisch

Umsatzsteuer-ID:

UST-ID: CH 583090

Kontakt:

Tel.: +41 (0) 41 7254210
Fax.: +41 (0) 41 7254211
E-Mail: kontakt@helpline-clm-ag.ch

Nutzungsbedingungen

Die in diesem Handbuch enthaltenen Angaben und Daten zur Software sind geistiges Eigentum von Serviceware und können ohne vorherige Ankündigung und ausschließlich von Serviceware geändert werden.

Kein Teil dieser Unterlage darf ohne ausdrückliche schriftliche Erlaubnis von Serviceware für irgendwelche Zwecke übertragen oder vervielfältigt werden. Das ist vollkommen unabhängig davon, auf welche Weise oder mit welchen Mitteln, ob dies elektronisch oder mechanisch geschieht. Obwohl bei der Zusammenstellung der Texte und Abbildungen mit größter Sorgfalt vorgegangen wurde, können Fehler nie vollkommen ausgeschlossen werden. Serviceware kann daher für fehlerhafte Angaben und ihre Folgen keine Haftung übernehmen. Serviceware geht mit diesem Dokument keine Verpflichtungen ein.

Die in den Beispielen verwendeten Unternehmens- und sonstigen Namen und Daten sind frei erfunden, soweit nichts anderes angegeben ist. Die Tatsache, dass in dieser Dokumentation Namen und Bezeichnungen genannt sind, begründet kein Recht auf freie Verwendung.

Stand: 4. August 2020

Inhaltsverzeichnis

1 Server Scripting

helpLine bietet Ihnen die Möglichkeit, mit Skripten zu arbeiten. Skripte können zum einen innerhalb des Dialog Designers in Dialoge eingebunden werden.

Genaue Informationen zum Verwenden von Skripten in Dialogen finden Sie im ***Administrationshandbuch Dialog Design***.

Außerdem können Skripte auf Server-Ebene genutzt werden, um Aktionen für helpLine Objekte auszuführen, die durch Aktionen des Benutzers ausgelöst werden. Diese Skripte werden im helpLine Designer verwaltet.

Dieser Teil des Handbuchs enthält einen Überblick über die Möglichkeiten des Scriptings in helpLine auf Server-Ebene sowie eine Referenz der zur Verfügung stehenden Schnittstellen und Methoden.

Server Scripting

1.1 Server-Skripte

In **helpLine Designer** finden Sie zu jedem Objekt links unten den Bereich „**EBL**“. Dort werden die Skripte für die in (unter „**Metainformationen**“/„**Objektdefinitionen**“) auf dessen Eigenschaften-Dialog die Registerkarte „**EBL**“, bei Vorgängen zusätzlich die Registerkarte „**Skript**“. Folgende Regeln gelten für sie:

- Sie gelten global für den Objekttyp bzw. die Assoziation.
- Es gibt keine Unterscheidung zwischen GUI und WEB oder zwischen unterschiedlichen Objekt-Dialogen.
- Es ist kein Zugriff auf Controls in Dialogen möglich.
- Drei feste Events sind festgelegt: OnCreate, OnSave, PostSave.
- Sie sind auch für Assoziationen möglich mit den Events „**OnCreate**“ und „**OnDelete**“.
- **Im Unterschied zu Dialogskripten und Delegations-/Fall-Skripten müssen bei Serverskripten obligatorisch alle Variablen deklariert werden („Dim“)!**

EBL (**E**xtended **B**usiness **L**ogic)-Skripte werden ausgeführt:

- beim Erzeugen einer Objektinstanz
- beim Speichern einer Objektinstanz
- nach dem Speichern einer Objektinstanz

Weitere Skripte werden ausgeführt:

- beim Erzeugen eines Falls
- beim Erzeugen eines delegierten Vorgangs
- beim Speichern eines delegierten Vorgangs

Im ClassicDesk kann zu Testzwecken die Ausführung von Server-(EBL-) Skripten durch Aktivierung des Kontrollkästchens „**Test Modus**“ unterdrückt werden (zu finden unter dem Menüpunkt „**Tools**“ >> „**Scripting**“).

Lesen Sie hierzu auch bitte das Kapitel „Skripte“ im *Administrationshandbuch helpLine Designer*.

1.1.1 EBL-Aktionen-Matrix

Mit nachfolgender Matrix bieten wir Ihnen einen Überblick darüber, welche potentiellen Aktionen in welchen EBL-Skripten möglich bzw. erlaubt sind:

Potentiell ausführbare Aktionen	Create	Save	Post Change	Hierarchy Changed
CurrentObject get AttributeValues	xe	xe	xe	-

Potentiell ausführbare Aktionen	Create	Save	Post Change	Hierarchy Changed
CurrentObject set AttributeValues (auch SU erstellen)	xe	xe	-	-
Abfrage RelatedObject (parent)	xe	xe	xe	-
RelatedObject get AttributeValues	xe	xe	xe	-
RelatedObject set AttributeValues (auch SU erstellen)	-	-	-	-
Abfrage des assoziierten Objekts	-	xe	xe	xe
AssocObject get AttributeValues	-	xe	xe	xe
AssocObject set AttributeValues (auch SU erstellen)	-	x	x	xesd
Abfrage assoziierter Objekte	-	xe	xe	xe
AssocObject get AttributeValues	-	xe	xe	xe
AssocObject set AttributeValues (auch SU erstellen)	-	-	x	xesd
Abbruch des Skriptes	xe	xe	x	xe
Ausführung von Templates auf CurrentObject	xe	-	-	-
Abfrage des AgentPerson Objektes	xe	xe	xe	xe
AgentPerson get AttributeValues	xe	xe	xe	xe
AgentPerson set AttributeValues	-	-	x	xesd
Setzen von Anfragern und Produkten	xe	-	-	-
Assoziationen erstellen	-	-	xe	-
Assoziationen löschen	-	x	x	-
SUCHEN (nach Objekten)	-	xe	xe	x
Get AttributeValues (der gefundenen Objekte)	-	xe	xe	x
Set AttributeValues (der gefundenen Objekte) (auch SU erstellen)	-	-	x	xsd
Create Object	-	x	xe	x
Save Object (NewObject's von CreateObject)	-	x	xe	x
Calls	-	x	xe	-
Rollback	xe	xe	-	xe

Zeichenerklärung:

xe: geht und ist erlaubt

x: geht, ist aber nicht empfohlen

-: ist verboten

xesd: geht und ist erlaubt, aber nur für Stammdaten

xsd: geht, ist aber nicht empfohlen und nur für Stammdaten

1.1.1.1 GetWaitingSupporters/GetControllingSupporters

Eine Funktion (aufrufbar im PostSave des Vorgangs) wurde hiermit bereitgestellt, die die Liste der Agenten zurückliefert, bei denen der Vorgang auf dem Tisch landet – und zwar unterteilt nach Warteschlange und Infotisch.

Beispiel:

```
Dim hlCase : Set hlCase = Nothing
Set hlCase = hlContext.GetCurrentObject
        Dim a1, WaitingSuporters, ControllingSuporters
        Dim arrW, arrC
        arrW = hlCase.GetWaitingSupporters
        arrC = hlCase.GetControllingSupporters
        a1 = a1 & " UBOUNDw:" & UBound(arrW)
        a1 = a1 & " UBOUNDc:" & UBound(arrC)
        For Each WaitingSuporters in arrW
                a1 = a1 & " w:" & WaitingSuporters
        Next
        For Each ControllingSuporters in arrC
                a1 = a1 & " c:" & ControllingSuporters
        Next
        hlCase.SetValue "CaseDiagnosis.DiagnosisText",0,0,0, a1

hlContext.trace 0, "EBL ONPOSTSAVE TRRIGER"
hlContext.trace 1, "EBL ONPOSTSAVE TRRIGER"
hlContext.trace 0, a1
hlContext.trace 1, a1
```

1.2 Delegations- und Fall-Skripte

Zentral auf dem Server gelten sie nur für Vorgangsobjekte (helpLine Administrator >> Eigenschaften des jeweiligen Objekttyps >> Registerkarte „**Skript**“):

Auch hier gibt es drei feste Events:

- Bei der Erzeugung eines Falls aus einem Vorgang
- Bei der Erzeugung eines delegierten Vorgangs
- Bei der Speicherung eines delegierten Vorgangs

1.3 Scripting Grundlagen

Über Skripte kann **auf alle Attribute eines Objekts innerhalb des Dialogs zugegriffen werden**. Bei Vorgangsobjekten kann zusätzlich auf Attribute des zugewiesenen Anfragers, der Organisationseinheit und des Produkts zugegriffen werden.

Um auf Vorgänge, Objekte, (deren) Attribute, Assoziationen usw. zugreifen zu können, müssen Sie Folgendes beachten:

- Im Unterschied zu **Dialog-** und **Delegations-/Fall-Skripten** müssen in den **Server-Skripten** obligatorisch alle Variablen deklariert werden („Dim“).

Hier weitere Grundlagen:

- Ein Abbruch des laufenden Speichervorgangs ist möglich, z. B. wenn kontextsensitiv fehlende oder ungültige Eingaben festgestellt wurden.
- In Dialogen werden GUI-Skripte über den Bereich „**Eigenschaften**“ (Property-Grid) im Dialog Designer eingebunden, WEB-Skripte über das Menü „**Ansicht**“ -> „**Skripte**“ im Dialog Designer.
- Server-Skripte werden auf dem Eigenschaftendialog der jeweiligen Objektdefinition oder Assoziation, auf der Registerkarte „**EBL**“ eingebunden.
- Delegations- und Fall-Skripte werden auf dem Eigenschaftendialog der jeweiligen Vorgangs-Objektdefinition, auf der Registerkarte „Skript“ eingebunden.
- Im ClassicDesk kann die Ausführung von Server-(EBL-)Skripten durch die Auswahl von „**Test Modus**“ unter dem Menüpunkt „Skripting“ unterdrückt werden (nur möglich mit der Sicherheitsrichtlinie „Globale Dialoge“).
- Es ist nicht möglich, von einem Skript aus ein anderes aufzurufen.

1.3.1 In EBL-Skripten für Objektdefinitionen

- hlcontext
- **hlcontext.GetRelatedObject** ermittelt das übergeordnete Objekt. Bei einer Neuanlage kann so z. B. ermittelt werden, unter welchem Objekt das neue Objekt angelegt wird.
- **hlcontext.GetCurrentObject** ermittelt das aktuelle Objekt.
- **hlcontext.GetScope** ermittelt, ob ein Vorgangsobjekt „normal“ angelegt wurde [liefert 0] oder ob es sich um eine Delegation [liefert 1] oder einen Fall [liefert 2] handelt.
- **hlcontext.AbortCommand <ErrorMessage>** bricht den laufenden Speichervorgang mit Fehlermeldung ab.
- **hlcontext.trace 1, <Description>** erzeugt einen Eintrag im Eventlog, in dem Text und Variablen übergeben werden können.
- **hlcontext.GetLocaleID** liefert die Locale/LCID der aktuellen Session.
- **hlcontext.GetLangIDFromLCID (<LCID>)** liefert die helpLine-Sprach-ID, um auf die Spracheinstellung des ClassicDesk zu reagieren.

- **hlcontext.GetAgentID ()** ermittelt die ID des angemeldeten Agenten.
- **hlcontext.GetPersonOfAgent (<AgentID>)** ermittelt den Namen des angemeldeten Agenten über dessen ID.
- **hlcontext.CreateObject (<ObjectType>)** erzeugt ein neues Objekt vom angegebenen Typ (Definitionsname oder -ID)
- **hlcontext.LoadObject (<ID>,<ObjectType>)** lädt ein Objekt über dessen Objekttyp und ID
- **hlcontext.SaveObject (<obj>)** speichert ein Objekt
- **hlcontext.DiscardChanges (<obj>)** verwirft alle Änderungen an den Objektdaten
- **hlcontext.RemoveObject (<obj>)** löscht ein Objekt
- **hlcontext.CreateAssociation (<ObjectA>,<ObjectB>,<AssocDef>)** erzeugt eine neue Assoziation zwischen dem übergeordneten ObjektA und dem untergeordneten ObjektB. <AssocDef> ist die Definitions-ID der Assoziation.
- **hlcontext.RemoveAssociation (<associd>,<assocdef>)** löscht die Assoziation zwischen dem übergeordneten ObjektA und dem untergeordneten ObjektB. <assocdef> ist die Definitions-ID oder der Definitionsname der Assoziation.
- **hlcontext.OpenSearch (<QueryString>)** sucht nach Objekten. Beispiel für <QueryString>: „SEARCH Employee WHERE HLOBJECTINFO.ID > 0“ findet alle Mitarbeiter. Syntax wie in der Expertensuche.
- **hlObj.GetTreeValuePath(AttributePath, LanguageId, ValueId)** Mit dieser Funktion lässt sich der „Langname“, bzw. der komplette Schlagwortpfad in EBL sprachabhängig für ein Schlagwort abfragen.
- **hlObj.GetTreeValueLevel(AttributePath, ValueId)** Mit dieser Funktion lässt sich die Ebene eines Schlagworts in EBL abfragen.

1.3.2 In EBL-Skripten für Assoziationen zusätzlich

- **hlobjectA** ermittelt das **übergeordnete** Objekt.
- **hlobjectB** ermittelt das **untergeordnete** Objekt.

1.3.3 In Delegations- und Fall-Skripten

- **hlcase** (alle Attribute des delegierenden bzw. fallbildenden Vorgangs)
- **hlcasefolder** (alle Attribute des übergeordneten Falls)
- **hldelegated** (alle Attribute des untergeordneten/delegierten Vorgangs)

Falls über das EBL-Skript keine neue Service-Einheit erzeugt wird, können die Attribut-Belegungen der aktuellen Service-Einheit überschrieben werden!

1.3.4 In Vorlagen und SLAs

- **hlobj** (alle Vorgangsattribute)

1.3.5 Connectivity

- **hlinquirer** (alle Anfragerattribute)
- **hlproduct** (alle Produktattribute)
- **hlsupporters** (alle Bearbeiterattribute)
- **hlcase** (alle Vorgangsattribute)
- **hlcasefolder** (alle Vorgangsattribute des übergeordneten Falls)

1.3.6 Control-unabhängige Befehle

- **GetValue("<ODE>.<Attribut>",a,b,c,d)** – liest ein Attribut aus.
  Beispiel: hlobj.GetValue("PersonGeneral.PersonName",0,0,0,0)
- **SetValue "<ODE>.<Attribut>",a,b,c,d** – schreibt einen Wert in ein Attribut.
  Beispiel: hlobj.SetValue "CaseAttribute.Priority",0,0,0,“PriorityHigh“

Obligatorisch zu übergebende Parameter:

a = Sprach-ID (7=Deutsch, 9=Englisch, 0=interner Name)

b = Content-ID (bei Multiple-Value-Attributen)

c = Index der Service-Einheit

d = Datentyp (0 = Definitionsname, 1 = Anzeigename) bzw. Wert, der gesetzt werden soll (Definitionsname!)

- **GetItems(&H10000,-1,-1,AssocDefID)** – liefert die über eine bestimmte Assoziation übergeordneten Objekte als Array.
  Beispiel: hlobj.GetItems(&H10000,-1,-1,103413) – liefert die einem Produkt übergeordneten Organisationseinheiten
- **GetItems(&H00000,-1,-1,AssocDefID)** – liefert die über eine bestimmte Assoziation untergeordneten Objekte als Array.
  Beispiel: hlobj.GetItems(&H00000,-1,-1,100703) – liefert die einem Mitarbeiter zugeordneten Produkte
- **GetItemCount (<flag>,<AssocDef>)** – ermittelt die Anzahl assoziierter Objekte. Flag wie oben.
  Beispiel: hlobj.GetItemCount (&H10000,103413) – liefert die Anzahl der einem Produkt übergeordneten Organisationseinheiten

- **AddItem <flag>,<Obj>,<AssocDef>** – ordnet das angegebene Objekt dem Objekt unter, für das die Methode aufgerufen wird.
  Beispiel: Orgunit.AddItem 0, Person, 103410 – ordnet das Personenobjekt <Person> der Organisationseinheit <Orgunit> zu.

- **RemoveItem <flag>,<Obj>,<AssocDef>** – löscht die Assoziation zwischen dem angegebenen untergeordneten Objekt und dem Objekt, für das die Methode aufgerufen wurde.
  Beispiel: Person.RemoveItem 0, Asset, 100703 löscht die Assoziation zwischen dem Personenobjekt <Person> und dem zugeordneten Inventar <Asset>.
- **GetType** – liefert den Objekttyp (Definitionsname).
  Beispiel: var = hlobj.GetType
- **GenerateID()** – generiert eine neue ContentID (für Mehrfachattribute)
- **GetLocale** – Setzt die LocaleID (LCID) für die aktuelle Session.
  Beispiel: lcid = hlContext.GetLocaleID
- **LangIDFromLCID** – ermittelt aus der LCID die Sprach-ID
  Beispiel: LangID = hlContext.LangIDFromLCID(lcid)

- **GetAttachmentKeys <Attribut>, <SUID>** – ermittelt die Inhalts-IDs von Anlagen.
  Beispiel: AttachIDs = hlCase.GetAttachmentKeys("HLOBJECTINFO.ATTACHMENT",0) ermittelt die Inhalts-IDs der Anlagen eines Vorgangs.
- **GetAttachment <Attribut>,<AttachID>, <SUID>** – ermittelt das Anlagenobjekt.
  Beispiel: Set hlAttachment=hlCase.GetAttachment ("HLOBJECTINFO.ATTACHMENT", AttachID, 0) ermittelt das Anlagenobjekt eines Vorgangs mit ID <AttachID>. Es kann dann mit Methoden wie **GetSize()**, **GetName()** weiter bearbeitet werden.

2 Der Script Context

Der ScriptContext ist das Ausgangsobjekt im **Server-Scripting**, über das auf alle anderen Objekte zugegriffen wird und das alle im Server-Scripting benötigten Methoden zur Verfügung stellt.

Alle in diesem Kapitel dargestellten Funktionen rufen Sie als Methode von „**hlcontext**“ auf:

Beispiel: hlcontext.GetCurrentObject bildet die Syntax zur Ermittlung des aktuellen Objektes.

Nicht alle aufgeführten Methoden stehen in allen Server-Skripten zur Verfügung (siehe dazu das Kapitel „*Objektmodell*“ im *Administrationshandbuch helpLine Designer*).

2.1 Session-Informationen

2.1.1 GetAgentID

„GetAgentID“ ermittelt die eindeutige Identifikationsnummer des angemeldeten Agenten über einen numerischen Wert.

Syntax: **GetAgentID ()**
Rückgabe: numerischer Wert

2.1.2 GetPersonOfAgent

„GetPersonAgent“ ermittelt die dem angegebenen Agenten zugeordnete Person.

Syntax: **GetPersonOfAgent (agent)**
Parameter: agent: numerischer Wert: ID des Agenten
Rückgabe: die dem Agenten zugeordnete Person als Variant (IHlObject)

GetPersonofAgent kann nicht im Postsave verwendet werden, da es nicht session-synchron ausgeführt wird.

2.1.3 GetLocaleID

„GetLocaleID“ ermittelt die lokale Identifikationsnummer oder auch Locale-ID (LCID) der aktuellen Session. Die LCID enthält neben der Sprach-ID auch Informationen zur Sortierung.

Syntax: **GetLocaleID ()**
Rückgabe: numerischer Wert

2.1.4 GetScope

„GetScope“ ermittelt den Kontext, in dem das Skript aufgerufen wurde. Abfrage nur im OnCreate-Skript verwendbar.

Syntax: **GetScope ()**
Rückgabe: 0: Standard; das Skript wurde bei einer „normalen“ Objektanlage aufgerufen
1: Delegation; das Skript wurde bei der Anlage eines neuen delegierten Vorgangs aufgerufen.
2: Fallerzeugung; das Skript wurde bei der Anlage eines neuen Falls aufgerufen.

2.1.5 IsInteractiveClient

„IsInteractiveClient“ ermittelt, ob der angemeldete Client interaktiv ist. Außer den Folgenden sind alle Clients interaktiv:

- helpLine Portal
- EAS
- Connectivity

Syntax: **IsInteractiveClient ()**
Rückgabe: boolean Wert („true“ oder „false“)

2.2 Skriptausführung

2.2.1 AbortCommand

„AbortCommand“ setzt einen Abbruchfehler. Zudem können Sie den Grund (*reason*) für den Abbruch des Skripts angeben. „AbortCommand“ bricht das Skript nicht selbst ab, die Ausführung des Skripts wird mit einer entsprechenden Anweisung wie „Exit Sub“ abgebrochen.

Syntax: **AbortCommand (reason)**
Parameter: reason
Rückgabe: Grund für den Abbruch des Skripts als String

2.2.2 Trace

„Trace“ gitb die für den Abbruch des Skripts angegebenen Beschreibung (*description*) aus. „Trace“ entspricht von der Funktionalität einer Message Box zur Ausgabe von Debug-Meldungen. Die Ausgabe der Beschreibung legen Sie über den Wert *flags* fest:

1. 1: Die Beschreibung wird als Information ins Eventlog eingetragen.
2. 2: Die Beschreibung wird als Information ins helpLine Log eingetragen.

Syntax: **Trace flags, description**
Parameter: flags, description
Flags: numerischer Wert:
1: Ausgabe ins Eventlog
2: Ausgabe ins helpLine Log.
Description: auszugebender Text als String

2.3 Objekt-Zugriff

2.3.1 GetCurrentObject

„GetCurrentObject“ ermittelt das von der Operation (Erzeugung, Speicherung) betroffene Objekt.

Syntax: **GetCurrentObject**
Rückgabe: das aktuelle Objekt als Variant (IHlObject)

2.3.2 GetRelatedObject

„GetRelatedObject“ ermittelt das dem aktuellen Objekt (s. o.) zugeordnete Objekt. Dabei kann es sich beispielsweise um dasjenige Objekt handeln, unter dem das aktuelle Objekt angelegt wird, oder um den Vorgang, zu dem ein anderer Vorgang assoziiert wird.

Das zugeordnete Objekt (RelatedObject) steht Ihnen nur bei den Aktionen „Anlegen“ und „Speichern“ zur Verfügung, nicht aber beim Event „PostSave“.

Syntax: **GetRelatedObject**
Rückgabe: das zugeordnete Objekt als Variant (IHlObject)

2.4 Objekte erstellen und bearbeiten

Mit den folgenden Methoden können Sie Objekte erzeugen, laden, speichern, löschen oder assoziieren.

2.4.1 CreateObject

„CreateObject“ erzeugt einen neuen Objekttyp vom angegebenen helpLine Typ.

Syntax: **CreateObject (ObjectType)**
Parameter: ObjectType: Symbolischer Name des gewünschten helpLine Objekttyps. Statt des Namens kann auch die Definitions-ID für den gewünschten Objekttyp angegeben werden.
Rückgabe erzeugtes Objekt als Variant (IHLObject)

2.4.2 LoadObject

„LoadObject“ lädt ein Objekt über dessen Objekttyp und ID.

Syntax **LoadObject (ID ,ObjectType)**
Parameter ID: Numerischer Wert
ObjectType: Definitionsname oder DefinitionID des gewünschten helpLine Objekttyps.
Rückgabe geladenes Objekt als Variant (IHLObject)

2.4.3 SaveObject

„SaveObject“ speichert das angegebene helpLine Objekt. Sie wird sowohl für das Speichern eines neuen Objekts als auch für das Speichern der Änderungen an einem bestehenden Objekt verwendet.

Syntax: SaveObject (Obj)
Parameter: Obj: Objekt, das in der Datenbank(en) gespeichert werden soll, als Variant (IHLObject)

Beispiel: Neues Objekt erzeugen und speichern
Neues Objekt vom Typ „Employee“ erzeugen und speichern. Der Nachname des (neuen) Mitarbeiters wird auf „Meier“ gesetzt.

```
dim NewEmpl
```

```
...
Set NewEmpl=hlcontext.createobject ("Employee")
NewEmpl.SetValue "PersonGeneral.PersonSurname", 0, 0, 0, "Meier"
hlcontext.saveobject NewEmpl
...
```

Beispiel 2: Bestehende Objekte ändern und Änderungen speichern.

Alle Objekte vom Typ „Employee“ mit dem Nachnamen (Attribut PersonSurname der ODE PersonalGeneral) „Neuer Mitarbeiter“ suchen, den Nachnamen auf „Meier“ ändern und speichern.

```
dim Empl
...
‘ Suche definieren
QryString = "SEARCH Employee WHERE PersonGeneral.PersonSurname =" _
                    & """Neuer Mitarbeiter"""
Set Qry = hlcontext.OpenSearch (QryString)
QryResult = Qry.GetItems(0, -1, -1, 0)
‘ Für jedes gefundenes Objekt Nachname ändern
‘ und Änderung speichern
For Each Empl in QryResult
 Empl.SetValue "PersonGeneral.PersonSurname", 0, 0, 0, "Meier"
 hlcontext.SaveObject(Emp)
Next ...
```

2.4.4 DiscardChanges

„DiscardChanges“ stellt den ursprünglichen Zustand des Objekts wieder her, d. h. die an Objektdaten vorgenommenen Änderungen werden verworfen.

Syntax: **DiscardChanges (object)**
Parameter: object: helpLine-Objekt als Variant (IHLObject)

2.4.5 RemoveObject

„RemoveObject“ löscht das angegebene helpLine Objekt und alle seine Assoziationen.

Syntax: **RemoveObject (Obj)**
Parameter: Obj: helpLine Objekt, das gelöscht werden soll als Variant (IHLObject)

Beispiel: Personen suchen und löschen

Alle Objekte vom Typ „Employee“ mit dem Nachnamen (Attribut „PersonSurname“ der ODE „PersonalGeneral“) „Meier“ löschen.

```
dim Empl

...

‘ Suche definieren

QryString = "SEARCH Employee WHERE PersonGeneral.PersonSurname =" _
```

```
                    & """Meier"""
Set Qry = hlcontext.OpenSearch (QryString)
QryResult = Qry.GetItems(0, -1, -1, 0)
‘ Für jedes gefundenes Objekt Nachname ändern
‘ und Änderung speichern
For Each Empl in QryResult
 hlcontext.RemoveObject (Empl)
Next
...
```

2.4.6 CreateAssociation

„CreateAssociation“ legt für die helpLine Objekte *ObjectA* und *ObjectB* eine Assoziation vom Typ *assocdef* an.

Syntax: **CreateAssociation (enda, endb)**

Parameter: ObjectA als Variant (IHLObject)

ObjectB als Variant (IHLObject)

Die Funktion liefert ab der helpLine Version 4.0 keine ID mehr zurück!

2.4.7 RemoveAssociation

„RemoveAssociation“ löscht die angegebene Assoziation mit der von „CreateAssociation“ zurückgegebenen Assoziations-ID.

Syntax: **RemoveAssociation (associd,assocdef)**

Parameter: **associd**: ID der zu löschenden Assoziation als numerischer Wert

assocdef: Typ der zu löschenden Assoziation als numerischer Wert

Da die Funktion „CreateAssociation“ ab der helpLine Version 4.0 keine ID mehr zurückliefert, wird an dieser Stelle empfohlen, „RemoveItem“ statt „RemoveAssociation“ zu verwenden.

2.5 Objekte suchen

2.5.1 OpenSearch

„OpenSearch“ sucht mittels der helpLine Expertensuche nach helpLine Objekten.

Syntax: **OpenSearch (QueryString)**

Parameter: QueryString: Suchbefehl als Variant (String)

Das Format entspricht der folgenden Form:

SEARCH Objekttyp WHERE Suchbedingung

	Objekttyp ist dabei der Definitionsname des gewünschten helpLine Objekttyps, beispielsweise „Employee“ oder „IncidentRequest“. Alternativ können Sie den Basistyp, für Vorgänge also CASE, angeben.
	Suchbedingung entspricht der Syntax der Experten-Suche
Rückgabe	Suchergebnis als Variant (IHLObject)

Beispiele zur Verwendung dieser Methode finden Sie auch in der Beschreibung der Methoden „SaveObject“ und „RemoveObject“.

Die Suchbedingung können Sie im helpLine ClassicDesk definieren und mit „Kopieren und Einfügen“ in Ihr Script kopieren.

Die Abfrage der einzelnen Objekte wird mit Hilfe der HLObject-Methode „GetItems“ erzielt. Über die HLObject-Methode „GetItemCount“ kann die Anzahl der gefundenen Objekte ermittelt werden. Die Werte der einzelnen Objekte werden mit Hilfe der HLObject-Methode „GetValue“ abgefragt. Dieses wird im folgenden Beispiel näher erläutert.

Beispiel: Liste der Unternehmen (Objekttyp Company) in einer Datei speichern.
Alle Objekte vom Typ „Company“ suchen und den Namen in eine Datei schreiben.

```
...
' Suche definieren
QryString = "SEARCH Company WHERE HLOBJECTINFO.ID >= 0"
Set Qry = hlcontext.OpenSearch (QryString)

' Ermitteln der Anzahl der gefundenen Objekte (Unternehmen)
objTextFile.Write "Unternehmen in der Liste: " & _
            Qry.GetItemCount (0, 0) & VbCrLf
objTextFile.Write "--------------------------------" & VbCrLf

' Namen der gefundenen Unternehmen in der Datei schreiben
Companies = Qry.GetItems (0, -1, -1, 0)
For each Company in Companies
  objTextFile.Write Company.GetValue _
            ("OrganisationGeneral.OrganisationName",0,0,0,0) & VbCrLf
Next

objTextFile.Write "--------------------------------" & VbCrLf
objTextFile.Close
```

2.6 Hilfsmethoden

2.6.1 GetDisplayName

„GetDisplayName“ ermittelt für die angegebene Locale-ID den Anzeigenamen eines helpLine Metadatenobjekts.

Syntax:	**GetDisplayName (ID,lcid)**
Parameter:	ID: ID des Metadatenobjekts als numerischer Wert (z. B. ID eines Attributs oder eine Objektdefinition)
	lcid: Locale-ID als numerischer Wert (z.B. 1031 für Deutsch, Deutschland oder 1033 für Englisch, USA)
Rückgabe:	Anzeigename als Variant (String)

2.6.2 GenerateID

„GenerateID“ generiert eine neue Content-ID (für Mehrfachattribute).

Syntax:	**GenerateID(type)**
	type: wird nicht verwendet
Rückgabe:	numerischer Wert

2.6.3 GetAttribute

„GetAttribute“ ermittelt das Attributobjekt für den angegebenen Schlüssel. Das Attributobjekt enthält Informationen über das angegebene Attribut, wie Typ, Status (z. B. ReadOnly) oder Vorbelegungen.

Syntax:	**GetAttribute (key)**
Parameter:	key
	Schlüssel des Attributs als Variant (String), beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME
Rückgabe:	Attributobjekt als Variant (IHLAttribute)

2.6.4 GetODEs

„GetODEs“ ermittelt die ODE-Objekte für die angegebene Objektdefinition

Syntax:	GetODEs(objdef)
Parameter:	objdef: Definitionsname der Objektdefinition, die die gewünschten ODEs enthält
Rückgabe:	ODEs der Objektdefintion als Variant (Collection, IODE)

2.6.5 GetLocale

„GetLocale“ ermittelt die Locale-ID (LCID) für die aktuelle Session. Die LCID enthält neben der Spracheinstellung auch Informationen zur Sortierung.

Syntax:	**GetLocale ()**
Rückgabe:	numerischer Wert
Beispiel:	**lcid=hlContext.GetLocaleID**

2.6.6 SetLocale

„SetLocale“ setzt die Locale-ID (LCID) für die aktuelle Session.

Syntax:	**SetLocale (lcid)**

Parameter: lcid: Locale-ID als numerischer Wert (z.B. 1031 für Deutsch, Deutschland oder 1033 für Englisch, USA)

2.6.7 GetAssociationChangesCount

„GetAssociationChangesCount“ ermittelt die Anzahl der Assoziationen, die geändert werden sollen.

Syntax: **GetAssociationChangesCount()**

Rückgabe: numerischer Wert

2.6.8 GetAssociationChangeAt

„GetAssociationChangeAt“ ermittelt das AssociationChange-Objekt zu dem angegebenen Index.

Syntax: **GetAssociationChangeAt(index)**

Parameter: index: Index einer zu ändernden Assoziation

Rückgabe: AssociationChange-Objekt als Variant

3 Allgemeine Schnittstellen

3.1 IHISession Interface

Über das IHlSession-Interface können Session-spezifische Daten abgefragt werden. Darüber hinaus stellt das IHlSession-Interface Hilfsmethoden für die Internationalisierung zur Verfügung.

Alle in diesem Unterkapitel dargestellten Funktionen rufen Sie als Methode von „**hlcontext**“ auf:

Beispiel: hlsession.GetAgentID ermittelt die eindeutige Identifikationsnummer des angemeldeten Agenten.

3.1.1 GetAgentID

„GetAgentID“ ermittelt die eindeutige Identifikationsnummer des angemeldeten Agenten.

Syntax: **GetAgentID ()**

Rückgabe: numerischer Wert

3.1.2 GetLocaleID

„GetLocaleID“ ermittelt die Locale-ID (LCID) der aktuellen Session. Die LCID enthält neben der Spracheinstellung auch Informationen zur Sortierung.

Syntax: **GetLocaleID ()**

Rückgabe: numerischer Wert

3.1.3 LangIDFromLCID

„LangIDFromLCID“ ermittelt aus der angegebenen Locale-ID (LCID) die primäre Sprach-ID, z. B. 7 für Deutsch und 9 für Englisch.

Syntax: **LangIDFromLCID (lcid)**

Parameter: lcid: numerischer Wert

Rückgabe: numerischer Wert

Beispiel: **LangID=hlContext.LangIDFromLCID(lcid)**

3.1.4 GetTranslation

„GetTranslation“ ermittelt für den angegebenen Schlüssel (z. B. Textfeldbezeichnung von Dialogfeldern) und die Locale-ID (LCID) die Übersetzung.

Syntax: **GetTranslation (key, lcid)**

Parameter: key: Schlüssel als Text (z.B. Eingabefeld_Ort).

lcid: numerischer Wert (z.B. 1031 für deutsch, Deutschland oder 1033 für englisch, USA).

Rückgabe: Text, sofern eine Übersetzung für den angegebenen Schlüssel in helpLine definiert ist. Ansonsten Rückgabe des Schlüssels selbst (z. B. Eingabefeld_Ort).

Die max. Länge für Übersetzungseinträge ist auf 2048 Zeichen begrenzt.

3.2 IhlObject-Interface

Über das IHlObject-Interface können Sie helpLine Objekte bearbeiten und deren Eigenschaften abfragen.

Alle in diesem Unterkapitel dargestellten Funktionen rufen Sie als Methode von „**hlobj**“ auf:

Beispiel: **hlobj.GetValue("PersonGeneral.PersonName",0,0,0,0) liefert den Namen zurück.**

3.2.1 Basisfunktionen

3.2.1.1 GetID

„GetID“ ermittelt die ID des Objekts.

Syntax: **GetID ()**

Rückgabe: numerischer Wert

3.2.1.2 GetType

„GetType“ ermittelt den Definitionsnamen des Objekts.

Syntax: **GetType ()**

Rückgabe: Definitionsname als Variant (String)

3.2.1.3 GetAttribute

„GetAttribute“ ermittelt das Attributobjekt für den angegebenen Schlüssel. Das Attributobjekt enthält Informationen über das angegebene Attribut, wie Typ, Status (z. B. ReadOnly) oder Vorbelegungen.

Syntax: **GetAttribute (key)**

Parameter: key: Schlüssel des Attributs als Variant (String), beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

Rückgabe: Attributobjekt als Variant (IHLAttribute)

3.2.1.4 IsNew

„ISNew“ prüft, ob das Objekt neu angelegt und noch nicht gespeichert wurde.

Syntax: **IsNew ()**

Rückgabe: numerischer Wert. Sofern das Objekt noch nicht gespeichert wurde, beträgt der Wert 1, andernfalls 0.

3.2.1.5 IsReadOnly

IsReadOnly“ prüft, ob das angegebene Attribut bearbeitet/geändert werden darf.

Syntax: **IsReadOnly (key, suidx)**

Parameter: *key*: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME. Wenn Sie einen leeren Schlüssel angeben, wird geprüft, ob das Objekt selbst geändert werden darf.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1 basierten Index der ServiceUnit an, für die das Attribut geprüft werden soll.

Bei Vorgängen greifen Sie über eine Belegung mit 0 auf die letzte Service-Einheit zu.

Sofern das Objekt nicht vom Typ Vorgang ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: numerischer Wert. Sofern das Attribut nicht geändert werden darf, beträgt der Wert 1, andernfalls 0.

Bitte beachten Sie, dass „**IsReadOnly**“ nur korrekte Werte zurückliefert, wenn das Recht „Besitzrechte übernehmen“ gesetzt ist.

3.2.1.6 Compare

„Compare“ vergleicht den Wert eines Attributs mit dem angegebenen Wert oder dem entsprechenden Attributwert eines anderen Objekts.

Syntax: **Compare (key, param, suidx)**

Parameter: key: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

param: Sofern Sie einen Attributwert mit einem bestimmten anderen Attributwert vergleichen möchten, muss dieser Wert als Text – Variant (String) – angegeben werden. Wenn Sie den Attributwert mit dem entsprechenden Wert eines anderen Objektes vergleichen möchten, muss dieses Objekt in param als Variant (IHlObject) angegeben werden.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die der Wert verglichen werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: numerischer Wert:

1: der Attributwert ist größer als der angegebene Wert

0: beide Werte sind gleich

-1: der Attributwert ist kleiner als der angegebene Wert

3.2.2 Zugriff auf Werte und Contents

3.2.2.1 GetValue

„GetValue“ ermittelt den Wert für das angegebene Attribut.

Syntax: **GetValue(*key*, *lcid*, *contentid*, *suidx*, *datatype*)**

Parameter: ***key***: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

lcid: numerischer Wert. Sprach-ID (z.B. 1031 für deutsch, Deutschland oder 1033 für englisch, USA)

contentid: numerischer Wert. Wird nur bei multiplen Attributen verwendet und gibt die Content-ID an, für die der Wert ermittelt werden soll.

Für alle anderen Attribute kann 0 angegeben werden.

Die ContentIDs multipler Attribute können über die Methode GetContentIDs ermittelt werden.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die der Wert ermittelt werden soll.

Sofern das Objekt nicht vom Typ Vorgang ist, können Sie den Wert mit einer 0 belegen.

datatype: numerischer Wert. Legt fest, welchen Datentyp der zurückgegebene Wert belegt.

Wert 0: Der Attributwert wird im Textformat zurückgegeben.

Wert 1: Der Attributwert wird in seinem ursprünglichen Format zurückgegeben, also Datum bei einem Datumsattribut, Text bei einem Zeichenkettenattribut usw.

Wert 2: Der komplette Schlagwortpfad wird zurückgegeben, z. B. „KeywordSoftware.KeywordOffice“.

Rückgabe: Variant, Wert. Der Typ des Variants hängt vom angegebenen datatype ab (s. o.).

Beispiel: **hlobj.GetValue("PersonGeneral.PersonName",0,0,0,0) liefert den Namen zurück.**

hlobj.GetValue("Keywords.Keyword",0,0,0,2) liefert das Keyword

3.2.2.2 GetTreeValuePath

Mit „GetTreeValuePath“ kann im Server-Skript der komplette Pfad ausgelesen werden.

Syntax: **GetTreeValuePath(*Keyword*, *langID*,*KeywordID*)**

Parameter ***Keyword***:Schlagwort;

langID, Sprach-ID z.B. 7 = Deutsch, 9 = Englisch

KeywordID: numerischer Wert

Beispiel:

```
Dim valKeywordID : valKeywordID = 0
Dim valKeywordIDs : valKeywordIDs = ""
Dim strTest : strTest = ""
valKeywordIDs = hlCase.GetValueIDs("Keywords.Keyword",0,0)
For Each valKeywordID IN valKeywordIDs
                    strTest = strTest & ", " &
hlCase.GetTreeValuePath("Keywords.Keyword",7,valKeywordID)
Next
```

3.2.2.3 SetValue

„SetValue“ setzt den Wert für das angegebene Attribut.

Syntax: **SetValue(*key*, *long lcid*, *contentid*, *suidx*, *value*)**

Parameter

key: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

lcid: numerischer Wert. Language-ID (z.B. 1031 für Deutsch, Deutschland oder 1033 für Englisch, USA)

contentid: numerischer Wert. Wird nur bei multiplen Attributen verwendet und gibt die Content-ID an, für die der Wert ermittelt werden soll.

Für alle anderen Attribute kann 0 angegeben werden.

Die ContentIDs multipler Attribute können über die Methode GetContentIDs ermittelt werden.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die der Wert gesetzt werden soll.

Sofern das Objekt nicht vom Typ Vorgang ist, können Sie den Wert mit einer 0 belegen.

Value: Variant (String) oder Wert, der für das angegebene Attribut gesetzt werden soll (Definitionsname)

Beispiel: **hlobj.SetValue "CaseClassificationAttribute.Priority",0,0,0,“PriorityHigh“ setzt die Priorität auf „hoch“.**

GetValue benötigt die Klammer, SetValue nicht.

3.2.2.4 HasContent

„HasContent“ prüft für multiple Attribute, ob Werte gesetzt sind.

Syntax: **HasContent (*key*, *reserved*, *suidx*)**

Parameter:

key: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME.

reserved: numerischer Wert, muss 0 sein.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1 basierten Index der ServiceUnit an, für die die Werte geprüft werden soll.

Bei Vorgängen greifen Sie über eine Belegung mit 0 auf die letzte Service-Einheit zu.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: numerischer Wert. Sofern das Attribut keinen Content aufweist, beträgt der Wert 0, andernfalls ist der Wert ungleich 0.

3.2.2.5 GetContentIDs

„GetContentIDs“ ermittelt die ContentIDs für multiple Attribute als Array. Mit „**GetValue“** kann über eine ContentID der entsprechende Wert abgefragt werden.

Syntax: **GetContentIDs (*key*, *suidx*)**

Parameter: ***key***: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1 basierten Index der ServiceUnit an, für die die ContentIDs ermittelt werden sollen.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: Variant (Collection, numerisch)

Beispiel: **ContIDs = hlCaller.GetContentIDs("PersonBilling.CostCenter_CA",0) ermittelt die Inhalts-IDs der Kostenstellen einer Person (des Anfragers).**

3.2.2.6 RemoveContentID

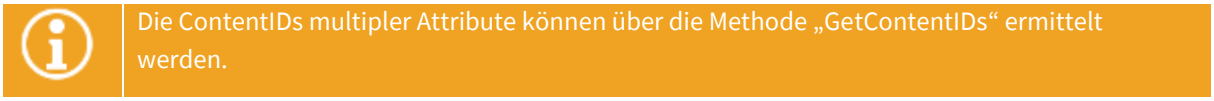
„RemoveContentID“ löscht bei einem multiplen Attribut (Mehrfachattribut) den angegebenen Wert (Datensatz).

Syntax: **RemoveContentID (*key*, *contentid*, *suidx*)**

Parameter: ***key***: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

contentid: numerischer Wert. Wird nur bei multiplen Attributen verwendet und gibt die Content-ID an, für die der Wert gelöscht werden soll.

Für alle anderen Attribute kann 0 angegeben werden.



suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die der Wert gelöscht werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

3.2.2.7 ResetContent

„ResetContent“ löscht alle Werte (Datensätze) eines multiplen Attributs.

Syntax: **ResetContent (*key*, *suidx*)**

Parameter: ***key***: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 0-basierten Index der Service-Einheit an, für die die Werte gelöscht werden sollen.

Sofern das Objekt nicht vom Typ Vorgang ist, können Sie den Wert mit einer 0 belegen.

3.2.2.8 GenerateContentID

„GenerateContentID“ erzeugt eine neue eindeutige Content-ID.

Syntax: **GenerateContentID()**

Rückgabe: numerischer Wert

3.2.2.9 SetValueIDs

„SetValueIDs“ setzt für ein multiples Attribut eine Liste von Werte-IDs.

Syntax: **SetValueIDs(*key*, *contentid*, *suidx*, *valueids*)**

Parameter: ***key***: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

contentid: numerischer Wert. Wird nur bei multiplen Attributen verwendet und gibt die Content-ID an, für die die Werte gesetzt werden sollen.

Für alle anderen Attribute kann 0 angegeben werden.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die der Wert gesetzt werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

Valueids: Werte-IDs als Variant (Collection, numerisch)

3.2.2.10 GetValueIDs

„GetValueIDs“ ermittelt die Werte-IDs eines multiplen Attributs.

Syntax: **GetValueIDs (*key*, *contentid*, *suidx*)**

Parameter: ***key***: Schlüssel des Attributs als Variant (String); beispielsweise die Anlegezeit des Objekts: HLOBJECTINFO.CREATIONTIME

contentid: numerischer Wert. Wird nur bei multiplen Attributen verwendet und gibt die Content-ID an, für die der Wert ermittelt werden soll.

Für alle anderen Attribute kann 0 angegeben werden.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die die Werte ermittelt werden sollen.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: Werte-IDs als Variant (Collection, numerisch)

3.2.2.11 GetLBText

„GetLBText“ liefert den Text des Listenelements mit dem angegebenen Index.

Syntax: **GetLBText (*idx*)**

Parameter: ***idx***: Index des Listenelements, von dem der Text zurückgegeben werden soll

Rückgabe: Text

3.2.3 Objektrelationen

3.2.3.1 GetItems

„GetItems“ ermittelt über die angegebene Assoziation die zu diesem Objekt assoziierten Objekte. Dabei können sowohl die untergeordneten als auch übergeordneten Objekte ermittelt werden.

Syntax: **GetItems (*flags*, *nfirst*, *nlast*, *assocdef*)**

Parameter:

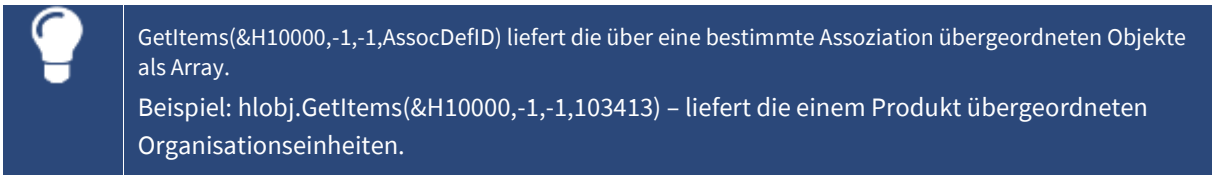
flags: numerischer Wert. Sofern Sie übergeordnete Objekte abfragen möchten, vergeben Sie den Wert „&H10000“, andernfalls (also für die untergeordneten Objekte) wählen Sie den Wert 0 („&H00000“).

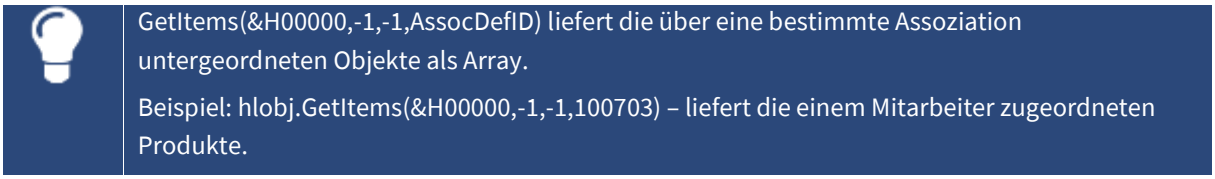
nfirst: numerischer Wert. Ein auf 0 basierender Index für das erste abzufragende Objekt. Sofern Sie alle Objekte abfragen möchten, geben Sie den Wert -1 an.

nlast: numerischer Wert. Ein auf 0 basierender Index für das letzte abzufragende Objekt. Sofern Sie alle Objekte abfragen möchten, geben Sie den Wert -1 an.

assocdef: Variant. Für diesen Wert kann entweder ein numerischer Wert, die Assoziations-Def-ID oder der Name der Assoziation (Variant(String)) angegeben werden.

Rückgabe: assoziierte helpLine Objekte als Variant (Collection, IHlObject)





3.2.3.2 GetItemCount

„GetItemCount“ ermittelt die Anzahl assoziierter Objekte. Für die angegebene Assoziation kann dabei die Anzahl sowohl für untergeordnete als auch übergeordnete Objekte ermittelt werden.

Syntax: **GetItemCount(*flags*, *assocdef*)**

Parameter:

flags: numerischer Wert. Sofern Sie übergeordnete Objekte abfragen möchten, vergeben Sie den Wert „&H10000“, andernfalls (also für die untergeordneten Objekte) wählen Sie den Wert 0 („&H00000“).

assocdef: Variant. Für diesen Wert kann entweder ein numerischer Wert, die Assoziations-Def-ID oder der Name der Assoziation (Variant(String)) angegeben werden.

Rückgabe: Anzahl der Objekte als numerischer Wert.

Beispiel: **hlobj.GetItemCount (&H10000,103413) liefert die Anzahl der einem Produkt übergeordneten Organisationseinheiten.**

3.2.3.3 AddItem

„AddItem“ ordnet das angegebene Objekt dem Objekt unter, für das diese Methode aufgerufen wird. Mit dieser Funktion können Sie Assoziationen objektbezogen verwalten.

Syntax: **AddItem (*flags*, *object*, *assocdef*)**

Parameter: ***flags***: numerischer Wert. Wird nicht verwendet und sollte 0 sein.

object: zuzuordnendes Objekt als Variant(IHlObject).

assocdef: Variant. Für diesen Wert kann entweder ein numerischer Wert, die Assoziations-Def-ID oder der Name der Assoziation (Variant (String)) angegeben werden.

Beispiel: **Orgunit.AddItem (0, Person, 103410) ordnet das Personenobjekt <Person> der Organisationseinheit <Orgunit> zu.**

3.2.3.4 RemoveItem

„RemoveItem“ löscht die angegebene Assoziation zwischen dem angegebenen untergeordneten Objekt und dem Objekt, für das diese Methode aufgerufen wird. Mit dieser Funktion können Sie Assoziationen objektbezogen verwalten.

Syntax: **RemoveItem (*flags*, *object*, *assocdef*)**

Parameter: ***flags***: numerischer Wert. Wird nicht verwendet und sollte 0 sein.

object: Objekt, zu dem die Assoziation gelöscht werden soll, als Variant(IHlObject)

assocdef: Variant. Für diesen Wert kann entweder ein numerischer Wert, die Assoziations-Def-ID oder der Name der Assoziation (Variant(String)) angegeben werden.

Beispiel: **Person.RemoveItem (0, Asset, 100703) löscht die Assoziation zwischen dem Personenobjekt <Person> und dem zugeordneten Inventar <Asset>.**

3.2.4 Anlagen

3.2.4.1 GetAttachmentKeys

„GetAttachmentKeys“ ermittelt für den angegebenen Attributschlüssel die IDs der an das Objekt angehängten Anlagen.

Syntax: **GetAttachmentKeys (*key*, *suidx*)**

Parameter: ***key***: Schlüssel des Anlagenattributs als Variant (String)

Der Standardschlüssel ist HLOBJECTINFO.ATTACHMENT.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die die IDs der Anlagen ermittelt werden sollen.

Sofern das Objekt nicht vom Typ Vorgang ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: Anlagen-IDs als Variant (Collection, numerisch). Das Anlagenobjekt für eine bestimmte ID kann mit der Methode „GetAttachment“ ermittelt werden.

Beispiel: **AttachIDs=hlCase.GetAttachmentKeys("HLOBJECTINFO.ATTACHMENT",0) ermittelt die Inhalts-IDs der Anlagen eines Vorgangs.**

3.2.4.2 GetAttachment

„GetAttachment“ ermittelt das Anlagenobjekt (Anlage) für die angegebene ID.

Syntax: **GetAttachment (*key*, *id*, *suidx*)**

Parameter: ***key***: Schlüssel des Anlagenattributs als Variant (String)

Der Standardschlüssel ist HLOBJECTINFO.ATTACHMENT.

id: Anlagen-ID als numerischer Wert. Die Anlagen-IDs können mit der Methode „GetAttachmentKeys“ ermittelt werden.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die die Anlage ermittelt werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: Anlagenobjekt als Variant(IHlAttachment)

Beispiel: **Set hlAttachment=hlCase.GetAttachment ("HLOBJECTINFO.ATTACHMENT", AttachID, 0) ermittelt das Anlagenobjekt eines Vorgangs mit ID <AttachID>. Es kann dann mit Methoden wie GetSize(), GetName() weiter bearbeitet werden.**

3.2.4.3 CreateAttachment

„CreateAttachment“ erzeugt eine neue Anlage.

Syntax: **CreateAttachment (*key*, *suidx*)**

Parameter: ***key***: Schlüssel des Anlagenattributs als Variant (String).

Der Standardschlüssel ist HLOBJECTINFO.ATTACHMENT.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die die Anlage angelegt werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

Rückgabe: Anlagenobjekt als Variant(IHlAttachment).

3.2.4.4 AppendAttachment

„AppendAttachment“ fügt dem Objekt eine neue Anlage hinzu.

Syntax: **AppendAttachment (*key*, *suidx*, *attachment*)**

Parameter: ***key***: Schlüssel des Anlagenattributs als Variant (String).

Der Standardschlüssel ist HLOBJECTINFO.ATTACHMENT.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die die Anlage hinzugefügt werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

attachment: Anlagenobjekt als Variant (IHlAttachment)

3.2.4.5 RemoveAttachment

„RemoveAttachment“ entfernt das angegebene Anlagenobjekt für die angegebene Service-Einheit aus dem Anlagenattribut.

Syntax: **RemoveAttachment (*key*, *suidx*, *attachment*)**

Parameter: ***key***: Schlüssel des Anlagenattributs als Variant (String).

Der Standardschlüssel ist HLOBJECTINFO.ATTACHMENT.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, aus der die Anlage entfernt werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

attachment: Anlagenobjekt als Variant (IHlAttachment)

Beispiel: **Entfernen aller Anlagen dieser Service-Einheit**:

```
Dim hlAttachKeys, hlAttachKey, hlAttach
hlAttachKeys=hlObj.GetAttachmentKeys("SUINFO.ATTACHMENT",model.CurrentSUIndex)
For Each hlAttachKey In hlAttachKeys
    Set hlAttach=hlObj.GetAttachment("SUINFO.ATTACHMENT",
    hlAttachKey,model.CurrentSUIndex)
    hlObj.RemoveAttachment "SUINFO.ATTACHMENT",model.CurrentSUIndex,
hlAttach
Next
```

3.2.4.6 SaveToFile

Mit „SaveToFile“ kann man in EBL- oder Connectivity-Skripten die helpline Anlagen im Dateisystem speichern.

Syntax: **strFilename = hlAttach.SaveToFile(0, "Wunschdateiname")**

Parameter: **hlAttach**: ist das Anlagenobjekt

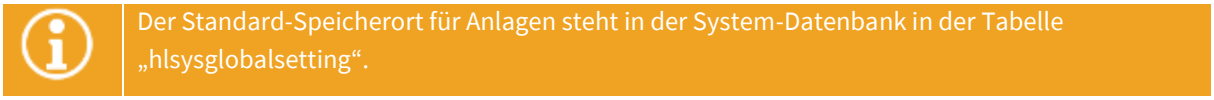
0 (erster Parameter): fix, nicht änderbar, aber erforderlich

Wunschdateiname: Dateiname, der verwendet werden soll, falls noch nicht vorhanden

strFilename: der „echte“ Dateiname, unter dem die Anlage abgelegt wird.

Dies ist wichtig, da sich der Dateiname ändert, wenn die Datei bereits vorhanden ist.

Beispiel: **Drei Vorgänge mit je einer Anlage „Neues Word-Dokument.doc“ (Wunschdateiname) werden exportiert. Man erhält anschließend die drei Dateien „Neues Word-Dokument.doc“, „Neues Word-Dokument (1).doc“ und „Neues Word-Dokument (2).doc“ (als „echte“ Dateinamen).**



Beispielskript:

Für Connectivity: Wenn eine Mail-Anlage zu groß ist, wird sie statt in helpLine im Dateisystem abgelegt und dies im Vorgang vermerkt.

```
Dim strURLOrig : strURLOrig = ""
Dim Size : Size = "Size: "
Dim attachmentsWarning : attachmentsWarning = ""
AttachCount = mail.CountAttachment
For nCount = 0 To AttachCount-1
 Set mailAttach = mail.Attachment(nCount)
 Size = Size & CStr(MailAttach.Size)
 'cut all attachments EXCEEDING LIMIT and add a WARNING MESSAGE on Problem field
 If MailAttach.Size > 25728640 Then ' In bytes z. B. 15728640
  attachmentsWarning = "WARNING MESSAGE: Attachments EXCEEDING LIMIT are cuted. Please check up your Mail Server Or contact your caller…"
 Else
  Dim hlAttach
  Set hlAttach = hlcase.CreateAttachment("HLOBJECTINFO.ATTACHMENT", 0)            'wenn CASE-Attachment
  hlAttach.url = MailAttach.Name
  If len(mailAttach.URL) > 0 Then
   hlAttach.Name = MailAttach.Name
   hlAttach.URL = mailAttach.URL
  Else
   'Namens-Anpassung des neuen Datei-Objekts (zusaetzlich Ref.Nr. + SUID),
   hlAttach.Name = strAttNameAddOn & MailAttach.Name
   'Dateiinhalt von Original-Mail-Anlage uebernehmen
   hlAttach.Data = MailAttach.Data
   'abspeichern im Dateisystem
   strURLOrig = hlAttach.SaveToFile(0, "")
   'INFO: Bei Serverumzug muss der Datenbankeintrag in der SysDB Tabelle
   'hlsysglobalsetting (= lok. oder UNC-PfadPfad inkl. abschl. Backslash <\> )
   'und der o. g. Pfad (strWebAddress) auf das HTTP- oder FTP-Verzeichnis angepasst werden.
   'Aendern des Pfades fuer den Remote-Zugriff (FTP oder http moeglich)
   'hierbei den neuen Pfad mit dem von der Funktion inkrementierten Objektnamen ergaenzen
```

```
  hlAttach.URL = strWebAddress & Mid(strURLOrig, instrRev(strURLOrig,"\")+1)
  End If
 End If
Next
```

3.2.5 Vorgangsspezifische Methoden

3.2.5.1 GetSvcUnitCount

„GetSvcUnitCount“ ermittelt die Anzahl der Service-Einheiten des Vorgangs.

Syntax: **GetSvcUnitCount ()**

Rückgabe: Anzahl der Service-Einheiten als numerischer Wert.

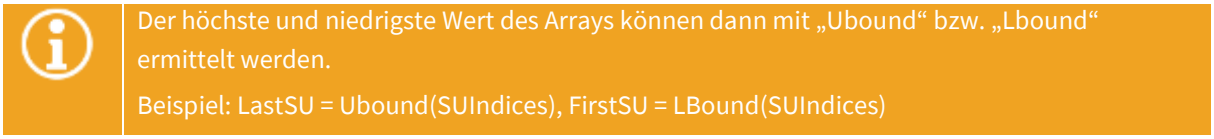
3.2.5.2 GetSvcUnitIndices

„GetSvcUnitIndices“ ermittelt die Indices der Service-Einheiten des Vorgangs und gibt sie als Array zurück.

Syntax: **GetSvcUnitIndices ()**

Rückgabe: Indices der Service-Einheiten als Variant (Collection, numerischer Wert)

Beispiel: **SUIndices = hlobj.GetSvcUnitIndices()**



3.2.5.3 AppendServiceUnit

„AppendServiceUnit“ hängt eine neue Service-Einheit an den Vorgang an.

Syntax: **AppendServiceUnit (flags)**

Parameter: flags: numerischer Wert. Wenn Sie Anfrager, Produkt und Organisation aus der vorherigen Servieeinheit übernehmen möchten, setzen Sie den Wert auf H1000, andernfalls auf 0.

Rückgabe: Index der neuen Service-Einheit als numerischer Wert.

3.2.5.4 GetReserver

„GetReserver“ ermittelt die ID des Agenten, auf die der Vorgang reserviert ist.

Syntax: **GetReserver ()**

Rückgabe: ID des Agenten als numerischer Wert

3.2.5.5 Reserve

„Reserve“ reserviert den Vorgang für den angemeldeten Agenten.

Syntax: **Reserve ()**

3.2.5.6 Unreserve

„Unreserve“ gibt den Vorgang frei.

Syntax: **Unreserve ()**

3.2.5.7 AddItemEx

„AddItem“ assoziiert das angegebene Objekt (object) entsprechend der angegebenen Assoziation zu dem Objekt, für das die Methode aufgerufen wird. Im Unterschied zur Methode „AddItem“ wird bei der Methode „AddItemEx“ auch der Service-Einheiten-Index mit angegeben, so dass diese Methode verwendet werden kann, um den Anfrager bzw. das Produkt einer Service-Einheit zu setzen.

Syntax: **AddItemEx (flags,object, suidx, assocdef)**

Parameter: flags: numerischer Wert (vgl. AddItem). Dieser Wert sollte immer 0 betragen.

object: zuzuordnendes Objekt als Variant(IHlObject)

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den auf 1 basierenden Index der Service-Einheit an, für die die Assoziation erstellt werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

assocdef: Variant. Für diesen Wert kann entweder ein numerischer Wert, die Assoziations-Def-ID oder der Name der Assoziation (Variant(String)) angegeben werden.

3.2.5.8 RemoveItemEx

„RemoveItem“ löscht die angegebene Assoziation zwischen dem angegebenen Objekt (object) und dem Objekt, für das die Methode aufgerufen wird. Im Unterschied zur Methode „RemoveItem“ wird bei der Methode „RemoveItemEx“ auch der Service-Einheiten-Index mit angegeben, so dass diese Methode verwendet werden kann, um Anfrager bzw. Produkt einer Service-Einheit zu entfernen.

Syntax: **RemoveItemEx (flags, object, suidx, assocdef)**

Parameter: flags: numerischer Wert (vgl. „RemoveItem“). Dieser Wert sollte immer 0 betragen.

object: zu entfernendes Objekt als Variant(IHlObject)

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die die Assoziation entfernt werden soll.

Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

assocdef: Variant. Für diesen Wert kann entweder ein numerischer Wert, die Assoziations-Def-ID oder der Name der Assoziation (Variant(String)) angegeben werden.

3.2.5.9 GetItemsEx

Es wird empfohlen, diese Methode mit Vorsicht zu verwenden, da hierbei alle assoziierten Objekte geladen werden und dies zu einer Verschlechterung der Performance führt.

„GetItemsEx“ ermittelt zu der angegebenen Assoziation die zu diesem Objekt assoziierten Objekte. Dabei können sowohl die untergeordneten als auch übergeordneten Objekte ermittelt werden. Im Unterschied zur Methode „GetItems“ wird bei der Methode „GetItemsEx“ auch der Service-Einheiten-Index mit angegeben, so dass diese Methode verwendet werden kann, um die zu einer Service-Einheit assoziierten Objekte (Anfrager, Produkt) abzufragen.

Syntax: **GetItemsEx (flags, suidx, assocdef)**

Parameter: flags: numerischer Wert (vgl. „GetItem“). Dieser Wert sollte immer 0 betragen.

suidx: numerischer Wert. Wird nur bei Vorgängen verwendet und gibt den 1-basierten Index der Service-Einheit an, für die die Objekte abgefragt werden sollen.

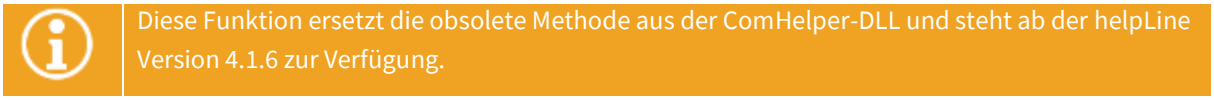
Sofern das Objekt nicht vom Typ „Vorgang“ ist, können Sie den Wert mit einer 0 belegen.

assocdef: Variant. Für diesen Wert kann entweder ein numerischer Wert, die Assoziations-Def-ID oder der Name der Assoziation (Variant(String)) angegeben werden.

Rückgabe: assoziierte helpLine Objekte als Variant (Collection, IHlObject)

3.2.5.9.1 GetPersonForAgent

„GetPersonForAgent“ liefert das dem angegebenen Agenten zugeordnete Personenobjekt zurück.



Syntax: **GetIPersonForAgent (AgentID)**

Parameter: AgentID: ID des Agenten

Rückgabe: Personenobjekt des Agenten

Beispiel:

Angemeldeten Agenten ermitteln

```
Dim AgentID : AgentID = 0
Dim Agent : Agent = ""
'Dim helper : Set helper = Nothing
Dim person : Set Person = Nothing
```

AgentID ermitteln

```
AgentID = hlSession.GetAgentID()
Verknüpfte Person ermitteln
Set helper = CreateObject("helpline.hlcontrols.HLHelperPFA")
If      isempty(helper.GetPersonForAgent (model.GetClientContext,AgentID)) = false then
        Set person = helper.GetPersonForAgent(model.GetClientContext,AgentID)
Nachnamen der verknüpften Person auslesen
```

msgbox **person.GetValue("PersonGeneral.PersonSurname",0,0,0,0) + ", " + person.GetValue ("PersonGeneral.PersonGivenName",0,0,0,0)**

else

msgbox "Es ist keine Person zugeordnet!", vbcritical

EndIf

3.3 IHlAttachment Interface

3.3.1 GetID

„GetID“ ermittelt die ID der Anlagen.

Syntax: **GetID ()**

Rückgabe: numerischer Wert

3.3.2 GetName

„GetName“ gibt den Namen der in dieser Anlage angehängten Datei zurück. Der Pfad einer Verknüpfung wird über „GetURL“ ermittelt.

Syntax: **GetName()**

Rückgabe: Name der angehängten Datei als Variant (String)

3.3.3 SetName

„SetName“ setzt den Namen in dieser Anlage.

Syntax: **SetName()**

3.3.4 GetURL

Wenn die Anlage eine Verknüpfung ist, wird durch „GetURL“ der vollständige Pfad (URL) auf die Anlage zurückgegeben. Für Dateianlagen ist der zurückgegebene String leer.

Syntax: **GetURL()**

Rückgabe: URL der Verknüpfung als Variant(String)

3.3.5 GetSize

„GetSize“ ermittelt die Größe der Anlage.

Syntax: **GetSize()**

Rückgabe: numerischer Wert. Die Größe der Anlage wird in Bytes angegeben. Für Verknüpfungen beträgt dieser Wert 0.

3.3.6 SetURL

„SetURL“ setzt die Verknüpfung zu einer Anlage. Angegeben werden muss der vollständige Pfad zur Anlage. Sie legen somit keine Kopie der Datei an, sondern speichern lediglich eine Referenz (Verknüpfung).

Durch „SetURL“ ersetzen Sie eine zuvor gesetzte Verknüpfung bzw. Dateianlage (SetFile).

Syntax: **SetURL (URL)**

Parameter: URL: vollständiger Pfad auf die Anlage als Variant(String)

3.3.7 SetFile

„SetFile“ setzt die Datei einer Anlage. Angegeben werden muss der vollständige Pfad auf die Anlage. Von der angegebenen Datei wird eine Kopie im helpLine-System gespeichert.

Durch SetFile ersetzen Sie eine zuvor gesetzte Dateianlage bzw. Verknüpfung (SetURL).

Syntax: **SetFile (filename)**

Parameter: filename: vollständiger Pfad auf die Anlage als Variant(String).

3.3.8 GetLastModified

„GetLastModified“ ermittelt, wann das Anlagenobjekt das letzte Mal geändert wurde.

Syntax: **GetLastModified()**

Rückgabe: Datum der letzten Änderung als Variant (Date)

3.3.9 SetData

„SetData“ setzt die Daten, mit denen die Anlage gespeichert werden soll.

Syntax: **SetData(ADODBStream, Dummy)**

Parameter: ADODBStream: Daten-Stream

Dummy: Dieser Wert sollte immer „0“ betragen.

3.3.10 GetData

„GetData“ gibt die Daten der Anlage zurück.

Syntax: **GetData(Stream)**

Parameter: Stream: Daten-Stream

3.4 IDefinition Interface

3.4.1 GetID

„GetID“ ermittelt die ID der Objektdefinition.

Syntax: **GetID ()**

Rückgabe: numerischer Wert

3.4.2 GetName

„GetName“ ermittelt den Anzeigenamen oder den internen Namen der Objektdefinition.

Syntax: **GetName(lcid)**

Parameter: lcid: Locale-ID als numerischer Wert (z.B. 1031 für Deutsch, Deutschland oder 1033 für Englisch, USA). Wenn Sie den internen Namen der Objektdefinition ermitteln möchten, geben Sie „0“ an.

Rückgabe: Anzeigename der Objektdefinition als Variant(String)

3.4.3 GetKey

„GetKey“ ermittelt den Definitionsnamen der Objektdefinition. Über den Definitionsnamen können die dieser Objektdefinition zugeordneten ODEs abgefragt werden. (s. a. context.GetODEs)

Für Basisdefinitionen wird der textuelle Bezeichner („CASE“, PRODUCT“, etc.) zurückgegeben.

Syntax: **GetKey ()**

Rückgabe: Definitionsname als Variant (String)

3.4.4 GetDefinitions

„GetDefinitions“ ermittelt die untergeordneten Definitionen.

Nur Basisdefinitionen wie „CASE“, „PRODUCT“, etc. haben untergeordnete Definitionen. „SupportRequest“ ist beispielsweise „CASE“ untergeordnet.

Syntax: **GetDefinitions()**

Rückgabe: untergeordnete Objektdefinitionen als Variant (Collection,IDefinition)

3.5 IODE Interface

3.5.1 GetID

„GetID“ ermittelt die ID der ODE.

Syntax: **GetID ()**

Rückgabe: numerischer Wert

3.5.2 GetName

„GetName“ ermittelt für die angegebene Sprache den Anzeigenamen oder den internen Namen der ODE.

Syntax: GetName(lcid)

Parameter: lcid: Locale-ID als numerischer Wert (z. B. 1031 für Deutsch, Deutschland oder 1033 für Englisch, USA). Wenn Sie den internen Namen der ODE ermitteln möchten, geben Sie „0“ an.

Rückgabe: Anzeigename der ODE als Variant(String).

3.5.3 GetKey

„GetKey“ ermittelt den Zugriffsschlüssel dieser ODE zu rein informativen Zwecken.

Syntax: **GetKey ()**

Rückgabe: Zugriffsschlüssel der ODE als Variant (String)

3.5.4 GetAttributes

„GetAttributes“ ermittelt die Attribute dieser ODE. Sie rufen damit alle Attribute ab, die der ODE zugeordnet sind.

Syntax: **GetAttributes()**

Rückgabe: Attribute der ODE als Variant (Collection, IHlAttribute)

3.6 IHlAttribute Interface

3.6.1 GetID

„GetID“ ermittelt die ID des Attributs.

Syntax: **GetID ()**

Rückgabe: numerischer Wert

3.6.2 GetName

„GetName“ ermittelt den Anzeigenamen oder den internen Namen des Attributs.

Syntax: **GetName(lcid)**

Parameter: lcid: Locale-ID als numerischer Wert (z. B. 1031 für deutsch, Deutschland oder 1033 für englisch, USA). Wenn Sie den internen Namen des Attributs ermitteln möchten, geben Sie „0“ an.

Rückgabe: Anzeigename des Attributs als Variant (String)

3.6.3 GetType

„GetType“ ermittelt den Objekttyp dieses Attributs. Der zurückgegebene Wert enthält neben den Informationen zum eigentlichen Attributtyp (numerisch, Datum, etc.) auch weitere Informationen zu den Attributeigenschaften wie ‚Hidden’, ‚ReadOnly’, ‚Multiple’, etc.

Syntax: **GetType ()**

Rückgabe: numerischer Wert

Beispiel: **var = hlobj.GetType**

Bedeutung der Rückgabewerte von IHLAttribute, GetType

Attributtyp	Wert
Unbekannt	&H00000000
Fließkomma	&H00100000
Numerisch	&H00200000
Text (varchar)	&H00300000
Text (lang)	&H00400000
Zeit	&H00500000
Bitfeld	&H00600000
Datum	&H00700000
DateTime	&H00800000
Währung	&H00900000
Dezimal Zahl	&H00A00000
URL	&H00B00000
Zeitspanne (von-bis)	&H00D00000
Zeitdauer	&H00E00000
Kennwort	&H00F00000

Eigenschaften	Wert
ReadOnly	&H01000000
Hidden	&H02000000
Required	&H04000000
Multiple	&H08000000
OneRequired (nur intern gebraucht)	&H10000000
Sorted (nur intern gebraucht)	&H20000000

Erweiterte Eigenschaften	Wert
Compound Attribut	&H00010000
List Attribut	&H00020000
Range Attribut	&H00030000
Tree Attribut	&H00040000

[1] Der zurückgegebene Wer setzt sich aus einer Kombination der oben aufgeführten Werte zusammen.

Beispiel:

Eine „GetType“-Abfrage auf ein ODE.Attribute liefert den (he-X) Wert &H2433000.

Dies bedeutet, dass das ODE.Attribute ein Range-Attribute vom Typ „Text“ (varchar) ist und zudem die Eigenschaften „Required“ und „Sorted“ hat:

	&H0003000	(RangeAttribute)
+	&H0030000	(Text varchar)
+	&H0400000	(Required)
+	&H2000000	(Sorted)
=	**&H2433000**	

3.6.4 GetKey

„GetKey“ ermittelt den Zugriffsschlüssel dieses Attributs.

Syntax: **GetKey ()**

Rückgabe: Zugriffsschlüssel des Attributs als Variant (String)

3.6.5 GetSubAttributes

„GetSubAttributes“ ermittelt diesem Attribut untergeordnete Attribute. Nur zusammengesetzte Attribute haben untergeordnete Attribute.

Syntax **GetSubAttributes ()**

Rückgabe untergeordnete Attribute als Variant (Collection, IHlAttribute).

3.6.6 GetContent

„GetContent“ ermittelt die Elemente für Baum- und Listenattribute

Syntax: **GetContent ()**

Rückgabe: Elemente als Variant (Collection, IContent)

3.6.7 IsChild

„IsChild“ ermittelt, ob ein Attribut einem anderen (anzugebenden) Attribut untergeordnet ist. Die Anwendung ist nur bei zusammengesetzten Attributen sinnvoll.

Syntax: **IsChild (attr)**

Parameter: attr: Die Attributkennung kann sein:

- die ID des Attributs (numerischer Wert)
- der Schlüssel des Attributs (Text)
- das Attribut selbst (IHlAttribute)

Rückgabe: numerischer Wert. Sofern das Attribut untergeordnet ist, erfolgt als Rückgabe der Wert „1“; andernfalls „0“.

Die Eigenschaft „GroupChildren“ ist im Dialog Designer nicht zu setzen, da die (Unter-)Attribute automatisch Children sind.

3.7 IContent Interface

Content-Objekte können hierarchisch strukturiert sein. Zur Abfrage solcher Strukturen steht analog zum Attribut-Interface die Methode **GetContent** zur Verfügung.

Bei Vorschlagslisten ist nur der über „GetName“ abzufragende Wert gültig.

3.7.1 GetID

„GetID“ ermittelt die ID für die Elemente von Baum- und Listenattributen.

Syntax: **GetID ()**

Rückgabe: numerischer Wert

3.7.2 GetName

„GetName“ ermittelt für die angegebene Sprache den Anzeigenamen oder den internen Namen des Contents.

Syntax: **GetName(lcid)**

Parameter: lcid: Locale-ID als numerischer Wert (z.B. 1031 für Deutsch, Deutschland oder 1033 für Englisch, USA). Wenn Sie den internen Namen des Contents ermitteln möchten, geben Sie „0“ an.

Rückgabe: Anzeigename des Contents als Variant (String)

3.7.3 GetContent

„GetContent“ ermittelt bei hierarchisch strukturierten Contents evtl. vorhandene untergeordnete Content-Objekte.

Syntax: **GetContent ()**

Rückgabe: untergeordnete Content-Objekte als Variant (Collection, IContent)

3.7.4 IsFrozen

„IsFrozen“ ermittelt, ob der Content eingefroren ist und damit nicht zur weiteren Verwendung zur Verfügung steht.

Syntax: **IsFrozen ()**

Rückgabe: numerischer Wert. Sofern der Content eingefroren ist, weist die Rückgabe den Wert „1“ auf, ansonsten „0“.

3.8 IHLEblAssociationChange Interface

In EBL-Skripten für Assoziationen zusätzlich:

- **hlcontext**
  - **hlobjectA** ermittelt das **übergeordnete** Objekt
  - **hlobjectB** ermittelt das **untergeordnete** Objekt

3.8.1 AssociationType

„AssociationType“ ermittelt den Namen der zu ändernden Assoziation.

Syntax: **AssociationType**

Rückgabe: der Name der Assoziation als Variant (String)

3.8.2 EndA

„EndA“ ermittelt das übergeordnete Objekt der Assoziation.

Syntax: **EndA**

Rückgabe: übergeordnetes Objekt als Variant (IHLObject)

3.8.3 EndB

„EndB“ ermittelt das untergeordnete Objekt der Assoziation.

Syntax: **EndB**

Rückgabe: untergeordnetes Objekt als Variant (IHLObject)

3.8.4 IsToCreate

„IsToCreate“ ermittelt, ob die Assoziation erzeugt werden soll.

Syntax: **IsToCreate()**

Rückgabe: boolean Wert (true oder false)

3.8.5 IsToDelete

„IsToDelete“ ermittelt, ob die Assoziation gelöscht werden soll.

Syntax: **IsToDelete()**

Rückgabe: boolean Wert (true oder false)

3.9 Connectivity

- **hlinquirer** (alle Anfragerattribute)
- **hlproduct** (alle Produktattribute)
- **hlsupporters** (alle Bearbeiterattribute)
- **hlcase** (alle Vorgangsattribute)
- **hlcasefolder** (alle Vorgangsattribute des übergeordneten Falls)

Neben den Skripten, die Sie im helpLine Administrator unter „**helpLine Dienste**“ -> „**Connectivity**“ -> „**Eigenschaften**“ -> „**EBL**“ angeben können, stehen Ihnen in helpLine noch ein neues Objekt und eine Methode zur Verfügung: „HLMail“ und „SendRequestMail“.

Bitte lesen Sie zur Installation und Konfiguration auch das *Administrationshandbuch Connectivity*.

helpLine bietet nun die Möglichkeit, via Serverskripting durch die Methode „SendRequestMail“ eine E-Mail zu initiieren. Der entsprechende Mailtext kann dabei im Serverskript individuell erstellt werden.

Das Objekt „HLMail“ hat die Eigenschaften:

- **To**
- **CC**
- **BCC**
- **Subject**
- **Body**
- **SenderMail**
- **Priority (0 = low, 1= normal, 2 = high) (default = 1)**

O. a. Methode funktioniert nur bei der Konfiguration der Connectivity für ausgehende Emails via SMTP – nicht bei CDO, da dort keine Absenderadresse geändert werden kann.

3.9.1 AddAttachment

„AddAttachment“ fügt der Mail eine Anlage hinzu.

AddAttachment([out, retval] VARIANT *pVal);

Syntax: **AddAttachment (attachment)**

Parameter: attachment: Anlage, die zu der Mail hinzugefügt werden soll als Variant (IHlAttachment)

Beispiel:

```
Dim mail
Set mail = hlContext.CreateMail
mail.To = "to@domain.de"
mail.CC = “cc@domain.de”
mail.BCC = “bcc@domain.de”
mail.SenderMail = “Absender@domain.de”
mail.Priority = 2
mail.Subject = "Request Mail"
mail.Body = "Hier wird der Text der Mail eingetragen"
Call hlContext.SendRequestMail(mail)
```

3.9.2 SendRequestMail

„SendRequestMail” schickt eine Mail per Connectivity.

Syntax: **SendRequestMail (mail)**

Parameter: mail: Mail, die verschickt werden soll (HLMail)

Beispiel:

```
dim mail
set mail = hlContext.CreateMail
mail.to = "Vorname.Nachname@pmcs.de"
mail.Subject = "Hello!"
mail.Body = "Nice to meet you!"
call hlContext.SendRequestMail (mail)
```

3.9.3 Notifydata.Notification

Über die Eigenschaft „Notifydata“ lässt sich herausfinden, durch welche konfigurierte Benachrichtigung eine Agentenbenachrichtigung ausgelöst wurde.

Syntax: **Notifydata.Notification**

Diese Eigenschaft steht nur im „AgentNotifyEmail“-Skript zur Verfügung.

Beispiel:

```
Dim notifyname
notifyname = Notifydata.Notification
if notifyname = „Reaktionszeitbedingung“ then
```

body = „Eine Reaktionszeitbedingung wurde erfüllt“

end if

3.9.4 Priority

Über das Attribut „Priority“ lässt sich die Priorität einer Mail setzen und abfragen.

Syntax **Mail.Priority**

Diese Eigenschaft steht nur in den Connectivity Skripten zur Verfügung.

Beispiel:

Dim prio

prio = mail.priority

if prio = 0 then ‘Priorität niedrig

if prio = 1 then ‘Priorität normal

if prio = 2 then ‘Priorität hoch

3.9.5 IsOnSupporterTable

Es wird geprüft, ob sich ein Vorgang auf einem Tisch des angegebenen Agenten befindet.

Syntax: **IsOnSupporterTable (SupporterName, table)**

Parameter: SupporterName (string): Hier muss der vollständige Name des Agenten angegeben warden.

Table (numerischer Wert): Hier wird angegeben, welcher Tisch geprüft werden soll. „4“ steht für Warteschlange, „10“ steht für Infotisch.

Rückgabe: boolean Wert, der angibt, ob sich der Vorgang auf dem Tisch des angegebenen Agenten befindet.

Beispiel:

If hlobj.IsOnSupporterTable(„Agenten Name“, 4) then

Body = “Der Vorgang ist in der Warteschlange”

End if

If hlobj.IsOnSupporterTable(„Agenten Name“, 10) then

Body = Body & “Der Vorgang ist im Infotisch”

End if

3.9.6 IsOnAnyTable

Es wird geprüft, ob sich ein Vorgang auf irgendeinem Tisch befindet.

Syntax: **IsOnAnyTable ()**

Rückgabe: **True:** Vorgang befindet sich auf irgendeinem Tisch

False: Vorgang befindet sich auf keinem Tisch

Beispiel:

```
if (false = hlobj.IsOnAnyTable) then
 ' befindet sich auf keinem Tisch
end if
```

3.9.7 GetWaitingSupporters/GetControllingSupporters

Hiermit wurde eine Funktion bereitgestellt, die die Liste der Agenten zurückliefert, bei denen der Vorgang auf dem Tisch landet – und zwar unterteilt nach Warteschlange und Infotisch.

Beispiel:

```
Dim a1, WaitingSuporters, ControllingSuporters
        Dim arrW, arrC
        arrW = hlcase.GetWaitingSupporters
        arrC = hlcase.GetControllingSupporters
        a1 = a1 & " UBOUNDw:" & UBound(arrW)
        a1 = a1 & " UBOUNDc:" & UBound(arrC)
        For Each WaitingSuporters in arrW
                a1 = a1 & " w:" & WaitingSuporters
        Next
                For Each ControllingSuporters in arrC
                a1 = a1 & " c:" & ControllingSuporters
        Next
        hlcase.SetValue "CaseDiagnosis.DiagnosisText",0,0,0, a1
```