Allgemeines

Die datenbanken24 SOAP Schnittstelle ermöglicht den automatisierten Zugriff auf Ihre datenbanken24.de Cloud-Application. Sie können über diese API:

- neue Dokumente erstellen
- bestehende Dokumente auslesen
- bestehende Dokumente bearbeiten bzw. ändern
- Dokumente löschen

Die Schnittstelle arbeitet als SOAP 1.1 RPC/encoded.
Alle Details dieser Schnittstelle können sie diesem Dokument oder den ebenfalls enthaltenen WSDL und XSD Dateien entnehmen.

Die WSDL Datei für Ihre Datenbank entnehmen Sie Ihrem Konfigurationsmenü: "SOAP API".
Oder Sie laden diese direkt über Ihre Datenbank-URL:
https://cloud-**xx**.datenbanken24.de/apps/**DBName**/base.nsf/DB24WebService?WSDL

SOAP-Version

SOAP 1.1 RPC/encoded
API Version vom 24.11.2015, benötigt datenbanken24 Version ab 10.37.19
Dokumentation, letzte Änderung: 11.11.2016

Lizenz

Der Zugriff auf Ihre datenbanken24 Cloud-Application per SOAP Schnittstelle ist kostenfrei, erfordert jedoch mindestens eine Business-Lizenz. Aus Sicherheitsgründen ist eine einmalige Freischaltung Ihrer Datenbank durch den MANETEC Support erforderlich.
Gerne stellen wir Ihnen auch eine Testumgebung zur Verfügung.
Kontaktieren Sie hierfür unsere Hotline.

Zugriffssicherheit und Authentifizierung

Die Schnittstelle arbeitet mit den Rechten des Benutzers, mit welchem sich diese über *setCredentials* authentifiziert. Hierfür können Sie jeden Benutzer aus Ihrer eigenen Benutzerverwaltung nutzen.

Es ist in den meisten Fällen sinnvoll,
einen Benutzer des Zugriffslevels "Manager" oder "Supervisor" anzugeben,
da diese Benutzer generelle Lese- und Schreibrechte auf **alle** Datensätze haben.

Soll die Schnittstelle explizit nur partielle Rechte auf eine Dokumentauswahl haben,
so kann diese gleichermaßen mit den Rechten eines "Autors" arbeiten.
Dokumente, die für diesen Benutzer nicht sichtbar sind, werden dann auch über die Schnittstelle nicht gefunden bzw. sind damit auch über die Schnittstelle nicht sichtbar.

Läuft Ihre Schnittstelle mit den Rechten eines "Lesers", so können Sie schreibende und löschende Funktionen der Schnittstelle mit diesem Benutzer nicht benutzen.

Datensatz = Dokument

datenbanken24.de ist eine nicht-relationale Dokumentendatenbank.

Ein datenbanken24 Dokument ist hierbei vergleichbar mit einem Datensatz / einem Record in einer relationalen Datenbank. Ein Dokument enthält bis zu 200 Felder und beschreibt z.B: ein Kundenprofil, einen Artikel, einen Bewerber, etc.

Ein datenbanken24 Formular entspricht grundsätzlich dem Aufbau einer Table.
Mit Formularen werden die nötigen Felder eines Dokuments individuell konfiguriert.
Für die Schnittstelle sind hierbei jedoch **nicht** Ihre individuell vergebenen Feldlabel relevant - sondern die internen Feldnummern der genutzten Felder.

Die Feldnummer eines Feldes erkennen Sie im Konfigurationsmenü "Formulargenerator" des entsprechenden Moduls. Hovern Sie mit der Maus über den Link "Feldeigenschaften".
Oder öffnen Sie den Feldeigenschaften-Dialog des betreffenden Feldes.
Dort steht im Reiter "Feldeigenschaften" z.B. der Hinweis:
Der interne Name dieses Feldes ist: F40.

Dieses Feld 40 können Sie in der Schnittstelle ansprechen über "FIELD_40"

Module umfassen alle Dokumente, die mit dem gleichem Formular erstellt wurden.
Sie sind vergleichbar mit Arbeitsmappen in Excel.
Die Zuordnung eines Dokuments zu einem Modul erfolgt über das Feld "ModuleNo",
in der Form "F1" bis "F10"

Initialisierung der Schnittstelle

Die WSDL Datei für Ihre Datenbank entnehmen Sie Ihrem Konfigurationsmenü: SOAP API.
Oder Sie laden diese direkt über Ihre Datenbank-URL:
https://cloud-**xx**.datenbanken24.de/apps/**DBName**/base.nsf/DB24WebService?WSDL

Diese WSDL können Sie als "Webservice-Consumer" in Ihre Entwicklungsumgebung importieren. In der WSDL ist bereits der Zielort Ihrer Cloud-Application bzw. Ihrer Datenbank enthalten.

Beispiel für eine Java Implementation

```
DB24APIService service = new DB24APIServiceLocator();
DominoSoapBindingStub db24 = (DominoSoapBindingStub)service.getDomino();
```

Beispiel für eine C# Implementation

```
var service = new DB24APIService();
```

Beispiel für eine php Implementation

```
ini_set("soap.wsdl_cache_enabled","0");


$user= 'Alfred Schmidt/my_crm';
$pass= 'password';

$client = new SoapClient("https://.../base.nsf/DB24WebService?WSDL",
  array('login' => $user,'password' => $pass));

$response = $client->testConnection();
```

Authentifizierung *ist Entwicklungsplattform-abhängig*

Die Authentifizierung erfolgt über eine spezielle Methode, die je nach Entwicklungsplattform unterschiedlich ist. Darum ist diese Funktion kein Bestandteil der datenbanken24 wsdl Datei und hier nicht direkt beschrieben.

Die Methode hat aber in allen Fällen Parameter für einen Benutzer und dessen Passwort. Wählen Sie einen Benutzer aus Ihrer Benutzerverwaltung aus, mit dessen Rechten die Schnittstelle auf Ihre Cloud-Application zugreifen darf.
In den allermeisten Fällen ist dies ein Benutzer des Typs "Manager" oder "Supervisor", da dieser Lese- und Schreibrechte auf **alle** Dokumente hat.

Beachten Sie:
Der Username Ihres Benutzers innerhalb der Schnittstelle setzt sich zusammen aus
Benutzername/Ihr Datenbankname

Ihren Datenbanknamen finden Sie in der Konfiguration Ihrer Datenbank
im Menü "Eigentümer & Lizenzvertrag" unter "Datenbankname".

Bsp.:
Ihr Benutzer heißt in der Benutzerverwaltung "Alfred Schmidt".
Der Name Ihrer Cloud-Application (Datenbankname) ist "my_crm"

Der Username Ihres Benutzers innerhalb der Schnittstelle wäre in diesem Fall:
Alfred Schmidt/my_crm

Beispiel für eine Java Implementation
db24.setCredentials("Alfred Schmidt/my_crm", "pw12345");

Sollte die Methode *setCredentials(String, String)* nicht vorhanden sein,
sollten Sie es mit
setUsername(String) **und** *setPassword(String)*
versuchen.

Beispiel für eine C# Implementation
Bei der C# Authentifizierung muss man folgenden Code in die automatisch erzeugte partielle Klasse DB24APIService einfügen:

```
protected override System.Net.WebRequest GetWebRequest(Uri Uri) {
        string loginData = "Basic " +
Convert.ToBase64String(System.Text.Encoding.UTF8.GetBytes("Alfred
Schmidt/my_crm:pw12345"));
        System.Net.HttpWebRequest request =
        (System.Net.HttpWebRequest)System.Net.HttpWebRequest.Create(this.Url);
        request.Headers["Authorization"] = loginData;
        return request;
}
```

Beispiel für eine php Implementation

```
$user= 'Alfred Schmidt/my_crm';
$pass= 'pw12345';

$client = new SoapClient("https://.../base.nsf/DB24WebService?WSDL",
  array('login' => $user,'password' => $pass));
```

Funktionen

Die hier beschriebenen Funktionen werden immer in der portType Klasse ***db24API*** bzw. im $Soap-Client Objekt aufgerufen.
In den Objekten ***db24API_Document*** und ***db24API_Module*** sind keine Funktionsaufrufe möglich. In diesen Objekten werden nur Properties gelesen und geschrieben.

Dokument-Update [bestehendes Dokument in Datenbank bearbeiten]

Der Ablauf einer Änderung bzw. eines Updates eines bestehenden Dokuments in Ihrer Datenbank wäre z.B.

1. Initialisieren des db24API_Document Objektes über eine der folgenden Funktionen (führt einen Webservice-Request durch)

   - *db24.getDocumentByID()*
   - *db24.getDocumentByIndKey()*
   - *db24.getDocumentBySearch()*

2. Ändern der Felder dieses Dokuments durch Überschreiben der Objekt-Eigenschaften (innerhalb Ihrer Entwicklungsumgebung, ohne Webservice-Request)

3. Senden des geänderten Dokument-Objektes an die Datenbank (führt einen Webservice-Request durch)

   db24.updateDocument(*db24Doc, $FieldArr)*

Neues Dokument [neues Dokument in Datenbank anlegen]

Der Ablauf für das Anlegen eines neuen Dokuments in Ihrer Datenbank wäre z.B.

1. Erzeugen eines neuen db24API_Document Objekts (innerhalb Ihrer Entwicklungsumgebung, ohne Webservice-Request)
   db24API_Document db24Doc = new db24API_Document();

   (oder z.B. für php, wenn keine SOAP-Objekte erzeugt werden können), durch
   $db24Doc = $client->getDocumentbyID("");
   Die Funktion "getDocumentbyID" mit Leerparameter ergibt ein leeres Dokument-Objekt.

2. Schreiben der Felder des neuen Dokuments durch Schreiben der Objekt-Eigenschaften (innerhalb Ihrer Entwicklungsumgebung, ohne Webservice-Request)

3. Senden des neuen Dokument-Objektes und des Field-Arrays an die Datenbank (führt einen Webservice-Request durch)

   db24.createDocument(*db24Doc, $FieldArr)*

testConnection()

Die Funktion *testConnection* ist dafür konzipiert, um in Ihrer Entwicklungsumgebung prüfen zu können, ob eine fehlerfreie Webservice-Verbindung zum datenbanken24 System aufgebaut werden kann. Sie wird nicht im Echtbetrieb benötigt - sondern ist als Hilfe während Ihrer eigenen Schnittstellen-Entwicklung und -programmierung ausgelegt.

Rückgabewert:
String. ”SUCCESSFULLY CONNECTED TO myCRM”
Im Falle einer fehlerfreien und funktionierenden Verbindung.
Ansonsten leer oder Fehlermeldung.

Beispiel für eine Java Implementation
String ret = db24.testConnection()

Beispiel für eine php Implementation
$response = $client->testConnection();

getAllDocumentIDs(ModuleNo)

Die Funktion *getAllDocumentIDs* gibt die Document-IDs von allen Dokumenten eines Moduls als String zurück.

Die Funktion ist für das programmatische ”Durchlaufen” aller Dokumente bzw. Datensätze eines Moduls konzipiert, wenn man Document-ID, Namen oder Werte von den in datenbanken24 gespeicherten Dokumenten nicht kennt.

Die Reihenfolge der Dokumente erfolgt quasi unsortiert. Sie entspricht technisch der alphanumerischen Sortierung der Document-ID.

Aus Kompatibilitätsgründen gibt die db24API kein Array sondern einen String zurück.
Die einzelnen IDs des Rückgabestring sind durch das Pipe-Zeichen | getrennt und können in Ihrer eigenen Entwicklungsumgebung leicht in ein Array umgewandelt werden.

Parameter
ModuleNo
String. Das datenbanken24-Modul, indem sich das gesuchte Dokument befindet.
F1, …, F10

Rückgabewert:
String. Document-IDs durch Pipe-Zeichen | getrennt

Beispiel für eine Java Implementation
String ret = db24.getAllDocumentIDs(*"F1")*

Beispiel für eine php Implementation
$ret = $client->getAllDocumentIDs("F1");

gibt alle Document-IDs des Moduls 1 (z.B. ”Kundenadressen”) in der Form zurück:
121018-31003-AD-927917928|121018-31003-AD-927917929|121018-31003-AD-927917930

getDocumentByID(Document-ID)

Die Funktion *getDocumentByID* sucht im gesamten Datenbestand nach einem Dokument mit der übergebenen internen Document-ID und gibt das gefundene Dokument als db24API_DOCUMENT Objekt zurück.

Parameter
Document-ID

String. Interne ID eines Dokuments.
Die Document-ID ist eine vom datenbanken24-System automatisch beim Anlegen eines neuen Dokuments erzeugte, eindeutige, permanente, alphanumerische ID.
Es kann jedoch auch ein konfiguriertes – individuelles - Feld zu dieser internen ID berechnet werden, z.B. wenn der Kunde sicherstellen kann, dass es sich bei diesem Feld um einen eindeutigen Wert handelt.

Den Wert des Feldes Document-ID finden Sie z.B. im Browser-Quellcode eines geöffneten Dokumentes in der Javascript-Variablen: DocID
var DocID = "121018-31003-AD-927917928";

Rückgabewert:
Objekt vom Typ: db24API_DOCUMENT

Beispiel für eine Java Implementation

```
db24API_Document db24Doc;
db24Doc = db24.getDocumentByID("121018-31003-AD-927917928");

System.out.println( db24Doc.getFIELD_1() ) ;
db24doc.setFIELD_1("Frau");
```

Beispiel für eine php Implementation

```
$db24Doc = $client->getDocumentbyID("121018-31003-AD-927917928");
```

getDocumentByIndKey(ModuleNo, FieldName, FieldValue)

Die Funktion *getDocumentByIndKey* sucht im Datenbestand des übergebenen Moduls (*ModuleNo*) nach einem Dokument, in welchem das Feld *Fieldname* den Wert *FieldValue* enthält.

Die Funktion ist als individuelle PrimaryKey Suche konzipiert. Sie können hiermit nach Dokumenten suchen, die mit eigenen PrimaryKey Feldern im jeweiligen Formular konfiguriert sind.

z.B.
Suche nach einem Kunden über Ihre eigene Kundennummer.
Suche nach einem Artikel über Ihre eigene Artikelnummer.

Parameter
ModuleNo
String. Das datenbanken24-Modul, indem sich das gesuchte Dokument befindet.
F1, …, F10

FieldName
String. Das eigene konfigurierte Feld, in dem der Feldwert gesucht wird.
FIELD_1, …, FIELD_199

FieldValue
String. Der Wert, der im Feld FieldName gesucht wird.

Rückgabewert:
Objekt vom Typ: db24API_DOCUMENT
Fragen Sie in diesem Dokument-Objekt die Document-ID ab,
ist diese leer, wurde kein Dokument mit dieser Abfrage gefunden,
hat diese einen Wert, wurde ein Dokument mit dieser Abfrage gefunden

Beispiel:

Im Modul "Kunden" (F1) haben Sie in Ihrem Feld "Kundennummer" (z.B. intern FIELD_12) die eigene Kundennummer Ihrer Kunden gespeichert.
Im Modul "Artikel" (F2) haben Sie in Ihrem Feld "Artikelnummer" (z.B. intern FIELD_82) die eigene Artikelnummer Ihrer Artikel gespeichert.

db24Doc = db24.getDocumentByIndKey("F1", "FIELD_12", "KN234")

gibt Ihnen das Dokument des Kunden mit der Kundennummer "KN234"
aus dem Modul "Kunden" als db24API_DOCUMENT zurück.

db24Doc = db24.getDocumentByIndKey("F2", "FIELD_82", "AN567")

gibt Ihnen das Dokument des Artikels mit der Artikelnummer "AN567"
aus dem Modul "Artikel" als db24API_DOCUMENT zurück.

Beispiel für eine php Implementation
$db24Doc = $client->getDocumentbyIndKey("F1", *"FIELD_12", "KN234")* ;
$Document-ID = $db24Doc->DOCID

updateDocument(*db24API_Document* , fieldsArray***)***

Die Funktion *updateDocument* schreibt die in Ihrem Field-Array durchgeführten Änderungen zurück in das datenbanken24-Dokument und speichert die Änderungen am Dokument in Ihrer Datenbank.

updateDocument nutzen Sie für bereits bestehende Dokumente
also wenn Sie das Dokument-Objekt z.B. über ein *getDocumentByID()* erzeugt haben.
Für neue Dokumente nutzen Sie hingegen die Funktion *createNewDocument().*

Parameter
Objekt vom Typ db24API_Document

Rückgabewert:
String. "1" = Success.
Anderenfalls enthält der String eine Fehlermeldung / ErrorMessage.

Beispiel für eine Java Implementation

String ret = db24.updateDocument(*db24Doc , Fields())*

Beispiel für eine php Implementation

```
$ret = $client->updateDocument($db24Doc, $fieldArr);
```

Das Beispiel schreibt alle am Dokument-Objekt vorgenommenen Änderungen per Webservice-Request zurück in Ihre Datenbank.

createNewDocument(db24API_Document , fieldsArray*)*

Die Funktion *createNewDocument* schreibt die in Ihrem Field-Array durchgeführten Feldzuweisungen in ein neues datenbanken24-Dokument und speichert das neue Dokument in Ihrer Datenbank.

createNewDocument nutzen Sie für neu zu erstellende Dokumente,
also dann, wenn Sie über die Schnittstelle neue Dokumente in Ihrer Datenbank anlegen.
Das db24API_Document hierfür erstellen Sie zuvor in Ihrer Entwicklungsumgebung über:

```
db24API_Document db24Doc = new db24API_Document();
```

Parameter
Objekt vom Typ db24API_Document

Rückgabewert:
String. Document-ID.
Gibt Ihnen die vom System erzeugte Document-ID für dieses neue Dokument zurück.
Anderenfalls enthält der String eine Fehlermeldung / ErrorMessage.

Beispiel für eine Java Implementation

```
db24API_Document db24Doc = new db24API_Document();
String DocID = db24.createNewDocument( db24Doc, Fields() )
```

Beispiel für eine php Implementation

```
$db24DocNeu = $client->getDocumentbyID("");
$db24DocNeu->MODULENO = "F2";
$DocID = $client->createNewDocument($db24DocNeu, $fieldArr);
```

Das Beispiel schreibt alle am Dokument-Objekt vorgenommenen Änderungen per Webservice-Request zurück in Ihre Datenbank.

deleteDocument(Document-ID)

Die Funktion *deleteDocument* sucht im gesamten Datenbestand nach einem Dokument mit der übergebenen internen Document-ID und löscht dieses permanent.

Parameter
Document-ID

String. Interne ID eines Dokuments.
Die Document-ID ist eine vom datenbanken24-System automatisch beim Anlegen eines neuen Dokuments erzeugte, eindeutige, permanente, alphanumerische ID.
Es kann jedoch auch ein konfiguriertes – individuelles - Feld zu dieser internen ID berechnet werden, z.B. wenn der Kunde sicherstellen kann, dass es sich bei diesem Feld um einen eindeutigen Wert handelt.

Den Wert des Feldes ID finden Sie z.B. im Browser-Quellcode eines geöffneten Dokumentes in der Javascript-Variablen: DocID
var DocID = "121018-31003-AD-927917928";

Rückgabewert:
String. "1" = Success.
Anderenfalls enthält der String eine Fehlermeldung / ErrorMessage.

Beispiel für eine Java Implementation

String ret = db24.deleteDocument(*"121018-31003-AD-927917928")*

Beispiel für eine php Implementation
$ret = $client->deleteDocument(*"121018-31003-AD-927917928")*

Das Beispiel löscht das Dokument mit der obigen ID permanent aus Ihrer Datenbank.

getFieldList(ModuleNo)

Die Funktion *getFieldList* gibt in Stringform eine Liste aller benutzten Felder dieses Moduls zurück.

Parameter
ModuleNo,
"F1" … "F10"

Rückgabewert:
String. Mit folgendem Aufbau (ohne den Zeilenumbruch, getrennt durch | Pipe-Zeichen)

F3=Name=TEXT|
F7=Straße=TEXT|
F23=Geburtsdatum=DATE|
…
F24=PLZ=NUMERIC

Beachten Sie:

Die angezeigten Typen (TEXT, DATE, NUMERIC) entsprechen der Konfiguration dieses Feldes im Formulargenerator. Dies sind lediglich Javascript Validierungsoptionen im Web-Formular.
Der Datentyp dieser Felder im db24API_DOCUMENT Objekt ist immer Text/ String.
und entspricht NICHT den in dieser Funktion angegebenen Typen.

Beim Setzen der Field-Properties **im db24API_DOCUMENT Objekt**
(also beim Schreiben von Feldern über die Schnittstelle)
sollte aber darauf geachtet werden, dass sich der Wert eines pseudo "numerischen" Feldes, dass Sie im Code als String in die Property schreiben -
innerhalb der datenbanken24 internen Verarbeitung dann in einen numerischen Wert umwandeln lässt. Anderenfalls wird der interne, wirklich numerische Wert als 0 gespeichert.

Beispiel für eine Java Implementation

String ret = db24.getFieldList("F3");

ret:
F3=Name=TEXT|F7=Straße=TEXT|F23=Geburtsdatum=DATE|F24=PLZ=NUMERIC

Beispiel für eine php Implementation

```
$fieldlist = $client->GETFIELDLIST("F1");
echo str_replace( "|", "<br />", $fieldlist);
```

updateDocumentField(Document-ID, Fieldname, FieldValue)

Die Funktion *updateDocumentField* ändert den Wert eines Einzelfeldes in einem bestimmten Dokument direkt über einen eigenen Webservices Request. Die Funktion sucht hierbei das Dokument über die Document-ID, ändert den Feldwert des betreffenden Feldes und speichert das Dokument sofort wieder.

Beachten Sie:
Jede Feldänderung bzw. jeder Aufruf dieser Funktion ist ein eigener Webservice-Request und benötigt somit Performance.

Wenn Sie mehrere Feldwerte eines Dokuments gleichzeitig ändern möchten, nutzen Sie anstelle von *updateDocumentField* besser ein db24API_DOCUMENT Objekt.

Parameter
Document-ID

String. Interne ID eines Dokuments.
Die Document-ID ist eine vom datenbanken24-System automatisch beim Anlegen eines neuen Dokuments erzeugte, eindeutige, permanente, alphanumerische ID.
Es kann jedoch auch ein konfiguriertes – individuelles - Feld zu dieser internen ID berechnet werden, z.B. wenn der Kunde sicherstellen kann, dass es sich bei diesem Feld um einen eindeutigen Wert handelt.

Den Wert des Feldes ID finden Sie z.B. im Browser-Quellcode eines geöffneten Dokumentes in der Javascript-Variablen: DocID
var DocID = "121018-31003-AD-927917928";

Fieldname
String. Das eigene konfigurierte Feld, dessen Feldwert geändert werden soll
FIELD_1, …, FIELD_199

FieldValue
String. Der Wert, mit dem das Feld Fieldname neu beschrieben werden soll.
Mehrfachwerte können durch | (Pipe-Zeichen) getrennt übergeben werden.

Rückgabewert:
String. "1" = Success.
Anderenfalls enthält der String eine Fehlermeldung / ErrorMessage.

Beispiel für eine Java Implementation

String ret= db24.updateDocumentField(*"121018-31003-AD-927917928", "FIELD_9", "Aktiv");*

Beispiel für eine php Implementation
$ret = $client->updateDocumentField(*"121018-31003-AD-927917928", "FIELD_9", "Aktiv");*

Das Beispiel setzt im Dokument mit der obigen ID den Feldwert des Feldes 9 auf "Aktiv".

getNumberOfDocuments(ModuleNo)

Die Funktion *getNumberOfDocuments* gibt Ihnen die Anzahl der Dokumente
Ihrer Cloud-Application zurück.
Mit einem ModuleNo-Parameter ("F1" … "F10")
ist dies die Anzahl der Dokumente in diesem Modul.
Mit einem Leerparameter ""
ist dies die Anzahl aller Dokumente in Ihrer gesamten Datenbank.

Rückgabewert mit Parameter "F1":
Numeric. Anzahl Dokumente im Modul 1 Ihrer Datenbank. z.B. Anzahl Kunden

Rückgabewert mit leerem Parameter "":
Numeric. Anzahl Dokumente in Ihrer gesamten Datenbank. z.B. Anzahl Kunden und Artikel

Beispiel für eine Java Implementation
Double ret = db24.getNumberOfDocuments("F1")

Beispiel für eine php Implementation
$double = $client->getNumberOfDocuments("F1")

getAllActiveModules()

Die Funktion *getAllActiveModules* gibt Ihnen die aktiven Module Ihrer Cloud-Application zurück.

Rückgabewert:
String, durch Komma getrennte Liste der aktiven Module,
z.B. "F1=Kunden|F3=Artikel|F5=Angebote"

Beispiel für eine Java Implementation
String ret = db24.getAllActiveModules()

Beispiel für eine php Implementation
$ret = $client->getAllActiveModules()

createRelation(Relation-ID, Document-ID, Document-ID2)

Die Funktion createRelation erstellt eine Relation(Verknüpfung) zwischen zwei Dokumenten.
Verknüpfungen des Typs 1:n, n:1 sowie m:n können erstellt werden.
Die Reihenfolge der IDs ist entspricht der Anzeige, wie es in der Konfiguration der Relationen eingestellt wurde.

Parameter
Die Relation-ID für die jeweilige Relation,
die Sie in Ihrem Konfigurationsmenü einsehen können.

String. ID der Relation.
Die Relation-ID befindet sich in der Konfiguration unter "Relationen".

Document-ID
FROM ID (im Relationen Menü das linke Modul)
String. Interne ID des Dokuments das verknüpft werden soll.

Document-ID2
TO ID (im Relationen Menü das rechte Modul)
String. Interne ID des Dokuments das verknüpft werden soll.

Rückgabewert:
String. Leer bei Erfolg.
Im Fehlerfall wird der Fehler zurückgegeben.

Beispiel für eine Java Implementation
String ret = db24.createRelation("M2M7-1n-M2S2", "121018-31003-AD-927917928", "121018-31003-AD-927917929");

Da datenbanken24.de eine nicht-relationale Dokumentendatenbank ist, muss das Aktualisieren der konfigurierten Referenzfelder (bei 1:n Verknüpfungen) explizit angestoßen werden.

Zum Aktualisieren der Referenzfelder nutzen Sie:
db24Doc = db24.getDocumentByID("121018-31003-AD-927917928");
db24.updateDocument(db24Doc , Fields())

Beispiel für eine php Implementation
$ret = $client->createRelation("M2M7-1n-M2S2", "121018-31003-AD-927917928", "121018-31003-AD-927917929");

Zum Aktualisieren der Referenzfelder (bei 1:n Verknüpfungen)
$db24Doc = $client->getDocumentbyID("121018-31003-AD-927917928");
$client->updateDocument($db24Doc, $fieldArr);

Das Objekt db24API_Document

Das Objekt db24API_Document bildet genau ein Dokument / einen Datensatz
aus Ihrer Datenbank ab.
Hierüber können Felder ausgelesen, geändert oder neu beschrieben werden.

Obwohl immer alle 200 Felder als Properties des *db24API_Document* Objektes in der API zur Verfügung stehen, werden beim Zurückspeichern des Objektes in das datenbanken24 System nur diejenigen Felder übernommen, die in der Konfiguration Ihres Formulars des jeweiligen Moduls enthalten sind - oder sich dort rechts in der "Parking area" befinden.

Das db24API_Document Object kann für bereits bestehende Dokumente in der Datenbank z.B. über *getDocumentByID(), getDocumentByIndKey(), getDocumentBySearch()* initialisiert werden.

Für neu zu erstellende Dokumente wird es über
set db24Doc = new db24API_Document()
erstellt.

Modul-Zuordnung von neuen Dokumenten

Die Zuordnung eines Dokuments zu einem Modul erfolgt über das Feld bzw. die Property "ModuleNo" in der Form "F1" ... "F10".

READONLY

Die Property DocID sowie die weiteren READONLY Properties können zwar innerhalb Ihrer Entwicklungsumgebung dem Objekt zugewiesen werden - werden aber nach dem Webservice Request auf Seiten von datenbanken24 NICHT ausgewertet bzw. NICHT zurück in das Dokument geschrieben.

Die Property FIELDS des Objekts db24API_Document

Die Property "FIELDS" gibt alle (maximal 200) Felder Ihres datenbanken24-Dokuments
als ein Array(0 - 200) zurück.

Dabei enthält das Element (0) als Information, alle aktiven Felder,
also diese Felder, die über die Konfiguration des Formulars überhaupt genutzt werden.
Nur diese Felder werden später beim Zurücksenden des Dokuments
vom datenbanken24-System als Änderung ausgewertet.

Dieses Array lesen, ändern oder bearbeiten Sie mit Ihrem eigenen Programmcode.
Änderungen an den Array-Elemente werden danach über die Methoden

*db24Doc.*updateDocument(DB24Doc, Fields())
*db24Doc.*createNewDoc(DB24Doc, Fields())

wieder zurück in das datenbanken24 Dokument geschrieben
oder in einem neuen datenbanken24 Dokument angelegt.

Feldwerte werden immer als String übergeben

Analog einer Eingabe im Webformular <input type=text>
werden auch über diese Schnittstelle alle Feldwerte im Textformat - also als String -
erwartet und übergeben. Bei Eingabe von Datumswerten oder numerischen Werten als String
beachten Sie bitte hierfür die im Konfigurationsmenü im Menü "Lokalisierung" festgelegten
Dezimaltrennzeichen und das eingestellte Kalenderformat, damit sich Ihre Feldwerte innerhalb
der datenbanken24 internen Verarbeitung dann in z.B. numerische Werte umwandeln lassen.

Mehrfachwerte

Die Schnittstelle nutzt aus Kompatibilitätsgründen keine verschachtelten Arrays für
Mehrfachwerte.
Mehrfachwerte (z.B. für Checkbox-Felder oder MultiValue-Auswahlfelder)
können über eine Verkettung der Werte mit dem Trenner "<mult>" übertragen werden.
In analoger Form werden diese auch im db24*API_Document* Objekt als Properties
bereitgestellt.

Wenn Sie in einem Checkbox-Feld, z.B die Einzelwerte
"Küche", "Bad", "Sat-TV", "Swimming-Pool", "Garage"
angeklickt haben oder über die Schnittstelle setzen möchten, dann erfolgt dies
sowohl lesend als auch schreibend in der Form:

"Küche<mult>Bad<mult>Sat-TV<mult>Swimming-Pool<mult>Garage"

Alias-Werte

Auswahllisten, die in Ihrem Formulargenerator als "Feld mit Alias-Werten" konfiguriert sind,
erwarten als Feldwert über die Schnittstelle den Alias-Wert.
Haben Sie eine Alias-Auswahl konfiguriert, etwa in der Form
männlich|M
weiblich|F
so muss die Schnittstelle den Alias-Wert "M" bzw. "F" übergeben.

Object db24API_Document: Properties

Property	Beschreibung	Datatype	Hinweis
DOCID	interne, eindeutige Dokument-ID	String	READONLY
MODULENO	F1, …, F10	String	
FIELDS	String Array(0 to 200) (0) = aktive Felder "|" (1) = Feld 1 (2) = Feld 2 (200) = Feld 200	Array of String	Mehrfachwerte innerhalb eines Feldes getrennt durch: <mult>
FIELD_HTML	Inhalt des HTML-Felder WYSIWYG-Funktionsfeld	String	
FIELD_TRIPLE1_1	Kaskadierende Auswahl 1		
FIELD_TRIPLE1_2	Hierarchie 2		
FIELD_TRIPLE1_3	Hierarchie 3		
FIELD_TRIPLE2_1	Kaskadierende Auswahl 2		
FIELD_TRIPLE2_2	Hierarchie 2		
FIELD_TRIPLE2_3	Hierarchie 3		
LCMT	Letzte Änderung am	String	READONLY
LCMU	Letzte Änderung durch	String	READONLY
ATTACHMENTCOUNT	Anzahl aller eingebettenden Dateianhänge in diesem Dokument	String	READONLY
ATTACHMENTNAMES	Name aller Dateianhänge in diesem Dokument	String "|"	READONLY
ATTACHMENTOVERALLSIZE	Gesamtgröße aller Dateianhänge	String [in Kb]	READONLY
CREATOR	Autor des Dokuments	String	READONLY
CREATIONTIME	Erstellzeit des Dokuments	String	READONLY
IMPORTID	ID des Import, mit welchem dieses Dokument erstellt wurde	String	READONLY

Code-Beispiel PHP

```
<?php

$datenbankname = "ihr datenbankname";
$cloudserver = "cloud-65.datenbanken24.de";  // z.B.: cloud-65.datenbanken24.de oder cloud-11.datenbanken24.de
$user = "ihr Benutzername";
$pw = "ihr password";

echo "PHP version: ".phpversion();

// initialisierung
ini_set("soap.wsdl_cache_enabled", "0");
ini_set('display_errors',1);
ini_set('display_startup_errors',1);
error_reporting(-1);

$userLogin = $user."/".$datenbankname;
$client = new SoapClient(
              "https://".$cloudserver."/apps/".$datenbankname."/base.nsf/DB24WebService?WSDL", array('login' => $userLogin,'password' => $pw
            ));

echo '<br><br>';

// testConnection
// Die Funktion testConnection ist dafür konzipiert, um in Ihrer Entwicklungsumgebung
// prüfen zu können, ob eine fehlerfreie Webservice-Verbindung zum datenbanken24 System aufgebaut werden kann.

$response = $client->testConnection();
echo "<b>Connection Test: </b><br>".$response;
echo '<br><br>';

// getAllDocumentIDs( ModuleNo )
// Die Funktion getAllDocumentIDs gibt die Dokumenten-IDs von allen Dokumenten eines Moduls als String zurück.
echo '<br><br>';

$alleDocIDs = $client->getAllDocumentIDs("F2");
echo "<b>Alle Dokumenten-IDs: </b><br>".$alleDocIDs;


// getDocumentByID( Document-ID )
// Die Funktion getDocumentByID sucht im gesamten Datenbestand nach einem Dokument
// mit der übergebenen internen Dokument-ID
// und gibt das gefundene Dokument als db24API_DOCUMENT Objekt zurück.
echo '<br><br>';

$DocIDsArray = explode( "|", $alleDocIDs);
$db24Doc = $client->getDocumentbyID($DocIDsArray[0]);
echo "<b>Modul: </b> ".$db24Doc->MODULENO ;


// getDocumentByIndKey( ModuleNo, Fieldname, FieldValue )
// Die Funktion getDocumentByIndKey sucht im Datenbestand des übergebenen Moduls (ModuleNo)
// nach einem Dokument, in welchem das Feld Fieldname den Wert FieldValue enthält.
echo '<br><br>';

$db24Doc = $client->getDocumentbyIndKey("F2", "FIELD_19", "Lamba") ;
$FieldArr = $db24Doc->FIELDS;
echo  "<b>Feld 18 aus dem Dokument: </b><br>". $FieldArr[18];

// Fragen Sie nun die Dokument-ID ab,
// ist diese leer, wurde kein Dokument mit dieser Abfrage gefunden
// hat diese einen Wert, wurde ein Dokument mit dieser Abfrage gefunden

echo  "<b>Dokument-ID aus dem Dokument: </b><br>". $db24Doc->DOCID;
echo '<br>';
echo  "<b>Feld 18 aus dem Dokument: </b><br>". $FieldArr[18];


// updateDocument( db24API_Document , fieldsArray)
// Die Funktion updateDocument schreibt die in Ihrem Field-Array durchgeführten Änderungen zurück
// in das datenbanken24-Dokument und speichert die Änderungen am Dokument in Ihrer Datenbank.
echo '<br><br>';

$FieldArr[18] = "Schnatterinchen";
$ret = $client->updateDocument($db24Doc, $FieldArr);
echo $ret."<br>";
echo "1 bedeutet, der Wert wurde erfolgreich im Dokument geändert";



// createNewDocument( db24API_Document , fieldsArray)
// Die Funktion createNewDocument schreibt die in Ihrem Field-Array durchgeführten Feldzuweisungen
// in ein neues datenbanken24-Dokument und speichert das neue Dokument in Ihrer Datenbank.
```

```
echo '<br><br>';

$db24DocNeu = $client->getDocumentbyID("");
$db24DocNeu->MODULENO = "F2";

$ret = $client->createNewDocument($db24DocNeu, $FieldArr);
echo "Es wurde ein neues Dokument mit der DocID: ".$ret." angelegt";



// deleteDocument( Document-ID )
// Die Funktion deleteDocument sucht im gesamten Datenbestand
// nach einem Dokument mit der übergebenen internen Dokument-ID und löscht dieses permanent.
echo "<br><br>";

$ret = $client->deleteDocument($ret);
echo $ret."<br>";
echo "1 bedeutet, das Dokument wurde erfolgreich gelöscht.";


// getFieldList( ModuleNo )
// Die Funktion getFieldList gibt in Stringform eine Liste aller benutzten Felder dieses Moduls zurück.
echo "<br><br>";

$fieldlist = $client->GETFIELDLIST("F1");
echo "<b>Alle konfigurierten Felder des Formulars des Moduls F1:</b><br>";
echo str_replace( "|", "<br />", $fieldlist);



// updateDocumentField( Document-ID, Fieldname, FieldValue )
// NICHT HÄUFIG bzw. SPARSAM BENUTZEN !
// Die Funktion updateDocField ändert den Wert eines Einzelfeldes in einem bestimmten Dokument
// direkt über einen eigenen Webservices Request.
// Die Funktion sucht hierbei das Dokument über die Dokument-ID,
// ändert den Feldwert des betreffenden Feldes und speichert das Dokument sofort wieder.
// Beachten Sie:
// Jede Feldänderung bzw. jeder Aufruf dieser Funktion ist ein eigener Webservice-Request und benötigt somit Performance.
echo "<br><br><b>Ändere das Feld 18 im Dokument mit der ID: ".$DocIDsArray[0]."</b>";

$ret = $client->updateDocumentField( $DocIDsArray[0], "FIELD_18", "Pittiplatsch");
echo $ret."<br>";
echo "1 bedeutet, der Wert wurde erfolgreich im Dokument geändert";



// getNumberOfDocuments(ModuleNo)
// Die Funktion getNumberOfDocuments gibt Ihnen die Anzahl der Dokumente
// Ihrer Cloud-Application zurück.
echo "<br><br><b>Anzahl aller Dokumente in der Datenbank: </b><br>";
echo $client->getNumberOfDocuments("");

echo "<br><br><b>Anzahl aller Dokumente im Modul 1: </b><br>";
echo $client->getNumberOfDocuments("F1");


// getAllActiveModules()
// Die Funktion getAllActiveModules gibt Ihnen die aktiven Module Ihrer Cloud-Application zurück.
echo "<br><br><b>Gibt die aktiven Module in der Datenbank zurück: </b><br>";
echo $client->getAllActiveModules();



// Die Property FIELDS des Objects db24API_Document
// Die Property "FIELDS" gibt alle (maximal 200) Felder Ihres datenbanken24-Dokuments
// als ein Array(0 - 200) zurück.
echo "<br><br><b>Field Array für das Dokument mit der ID: ".$DocIDsArray[0]."</b><br><br>";

$db24Doc = $client->getDocumentbyID($DocIDsArray[0]);
$fieldarray = $db24Doc->FIELDS;

foreach ($fieldarray as $key => $value) {
echo "Feld: ".$key." = ".$value."<br />\n";
};

 ?>
```

Code-Beispiel Java

```
import DefaultNamespace.*;

public class Program {

    public static void main(String[] args) {

        try {

            DB24APIServiceLocator serviceLocator = new DB24APIServiceLocator();
            DominoSoapBindingStub db24 = (DominoSoapBindingStub)serviceLocator.getDomino();
            db24.setCredentials("Benutzername/Datenbankname", "Passwort");

            // Wenn setCredentials nicht existiert:
            // db24.setUsername("Benutzername/Datenbankname");
            // db24.setPassword("Passwort");

            String test = db24.TESTCONNECTION();
            System.out.println(test);

        } catch(Exception exception) {

            exception.printStackTrace();
            System.exit(1);
        }
    }
}
```

Wenn dieser Beispiel-Code (Erreichen einer Connection) den Wert „Success" ausgibt – konnte die Verbindung erfolgreich aufgebaut werden.
Wenn „Failure" ausgegeben wird, konnte keine Verbindung aufgebaut werden.