Arbeiten mit Kalendereinträgen

AID 029a DE

© 2016 ADITO Software GmbH

Diese Unterlagen wurden mit größtmöglicher Sorgfalt hergestellt. Dennoch kann für Fehler in den Beschreibungen und Erklärungen keine Haftung übernommen werden. Wir sind für Feedback zu den Themen, Inhalten, aber auch noch vorhandenen Fehlern dankbar und würden uns freuen, Ihre Meinung zu hören. Die in diesen Unterlagen enthaltenen Daten und Angaben, einschließlich URLs und anderer Verweise können ohne vorherige Ankündigung geändert werden. Alle in diesen Unterlagen aufgeführten Produkt- und Firmennamen sind unter Umständen Marken oder geschützte Zeichen der einzelnen Firmen. Ohne ausdrückliche schriftliche Einverständniserklärung der ADITO Software GmbH darf kein Teil dieses Dokumentes vervielfältigt oder in einer Datenverarbeitungsanlage gespeichert oder in diese eingelesen werden. Diese Einschränkung gilt unabhängig von Art und Weise der Datenerfassung.

Autor: FA, KN, MW. Version 10.2. Zuletzt geändert 04.09.2017

Version	Bemerkung
10.2	Anpassung der Formatierungen
10.1	Beispiele überarbeitet
10.0	Anpassung an ADITO4.6
4.3	Anpassung des Formats
4.2	Letzte Version vor Umstellung der Versionshistorie im Dokument

Inhaltsverzeichnis

1. Kalenderbenutzer und Rechte

Bei Start des ADITO4 Clients wird im Prozess `autostart` der Kalender-Benutzer über die JDito-Funktion `calendars.setCalendarUser` gesetzt. Der Kalenderbenutzer ist also nicht fest mit dem jeweiligen ADITO-Benutzer verbunden, sondern wird zur Laufzeit gesetzt.

1.1. Kalenderrechte

Im ADITO-Kalender gibt es zwei Kategorien von Rechten, die gesetzt werden können:

- Leserechte
- Schreibrechte

Ist im ADITO-Kalender das "Leserecht" gesetzt, so steht dem Benutzer immer das vollständige Kalenderobjekt zur Verfügung. Eine Anzeige mit weniger Details wie beispielsweise in Microsoft Exchange existiert in ADITO nicht.

Diese Rechte können auf verschiedenen Arten gesetzt werden:

- Rechte für alle Kalenderelemente (Aufgaben und Termine)
- Rechte für Aufgaben
- Rechte für Termine

Sind keine Rechte explizit für Aufgaben oder Termine gesetzt, gelten die allgemeinen Berechtigungen.

Schaubild dazu:

	Allgemeines Recht Gilt wenn → nicht gesetzt	**Spezifisches Recht Aufgaben**	**Spezifisches Recht Termine**
Leserechte	`calendars.RIGHT_READ`	`calendars.RIGHT_READ_TASK`	`calendars.RIGHT_READ_APPOINTMENT`
Schreibrechte	`calendars.RIGHT_WRITE`	`calendars.RIGHT_WRITE_TASK`	`calendars.RIGHT_WRITE_APPOINTMENT`

1.2. Rechte in Verbindung mit anderen Backends

Besitzt ein angeschlossenes Kalenderbackend eine andere Rechtestruktur wie ADITO, so werden die Rechte des Backends auf die jeweils strengere Stufe in ADITO übernommen. Das bedeutet beispielsweise bei Microsoft Exchange, dass das Leserecht "Eingeschränkte Details", welches die Ansicht von Datum und Betreff gestattet, in "Kein Leserecht" umgewandelt wird.

Ist aber Kalendersynchronisation und Kalendercaching aktiviert, dann gelten beim Zugriff auf die im ADITO-Datenbankkalender befindlichen Elemente wieder die ADITO-Rechte.

2. Neuer Kalendereintrag

2.1. Aufbau eines Kalendereintrags

Jeder Kalendereintrag besitzt die in der Tabelle unten angegeben Felder. Einige dieser Felder sind nur für Termine, einige nur für Aufgaben verfügbar (siehe jeweiligen Eintrag).

Feld	Beschreibung
AFFECTEDUSERS	Beteiligte Benutzer als Array von Kalenderbenutzern.
ATTENDEES	Teilnehmer als Array von Kalenderbenutzern.
CATEGORIES	Liefert die Kategorien für einen Eintrag.
CLASSIFICATION	Klassifizierung (CLASSIFICATION_PRIVATE oder CLASSIFICATION_PUBLIC).
COMMENT	Kommentar
CREATED	Erstellungsdatum des Eintrags.
DESCRIPTION	Beschreibung für einen Termin bzw. eine Aufgabe.
DTEND	Enddatum eines Termins.
DTSTART	Startdatum
DTSTAMP	Zeitstempel
DUE	Fälligkeitsdatum einer Aufgabe.
DURATION	Dauer, statt DTEND oder DUE.
EXDATE	Daten der gelöschten Elemente einer Terminserie.
FREEBUSY	Frei / Gebucht
HASREMINDER	Hat dieser Eintrag eine Erinnerung? ("true" oder "false").
ID	Die Identifikation des Kalendereintrags (UID).
LASTMODIFIED	Datum der letzten Änderung.
LINKS	Verknüpfungen für diesen Eintrag (siehe unten).
LOCATION	Der Ort für den Termin.
ORGANIZER	Der Ersteller eines Kalendereintrags (Login des Users).
ORGANIZER2	Der Ersteller eines Kalendereintrages (vollständiger Name incl. ID und CN).

PERCENT	Fertigstellungsgrad in Prozent der Aufgabe.
PRIORITY	Priorität des Eintrags (0=undefiniert, 1=hoch bis 9=niedrig).
RECURRENCEID	Verweis auf den Haupteintrag für eine Serienausnahme.
RRULE	Definition der Wiederholung einer Terminserie.
RDATE	Daten für die Terminserie.
REMINDER	Zeitpunkt der Erinnerung als Timestamp.
STATUS	Statuskennung des Eintrags (s. Statusliste unten).
SUMMARY	Betreff des Eintrags bzw. Name der Aufgabe.
TYPE	Der Typ des Kalendereintrags (`calendars.VEVENT` oder `calendars.VTODO`).
USER	Der Eigentümer des Kalendereintrags (wer ist zuständig).
USER2	Der Eigentümer des Kalendereintrags (vollständiger Name incl. ID und CN).
RIGHT_READ	Leserecht
RIGHT_READ_APPOINTMENT	Leserecht für Termine.
RIGHT_READ_TASK	Leserecht für Aufgaben.
RIGHT_WRITE	Schreibrecht
RIGHT_WRITE_APPOINTMENT	Schreibrecht für Termine.
RIGHT_WRITE_TASK	Schreibrecht für Aufgaben.

2.2. Kalendereintrag anlegen

Ein Kalendereintrag wird als JavaScript-Map aufgebaut und dann mit der Methode `calendars.insert` im Kalender-Backend gespeichert oder mit der Methode `swing.openCalendarEntry` / `swing.openCalendarEntryByUID` zum Bearbeiten geöffnet.

Erzeugen Sie zunächst eine leere Map (Array).

```
var entry = new Array();
```

Diese Map befüllen Sie mit den Feldern des Kalendereintrags, z.B.:

```
entry[calendars.TYPE] = calendars.VTODO;

…
```

Das so erzeugte Objekt stellt eine Aufgabe, einen einfachen Termin oder die Seriendefinition eines Serientermins dar. Dieses Objekt nimmt die erste Stelle eines Objekt-Arrays ein, das

ein Kalenderobjekt beschreibt. Im Fall von Serienterminen kann dieses Objekt-Array weitere Einträge, die die Ausnahmen der Terminserie beschreiben, enthalten.

Dieses Objekt-Array wird nun mit der Methode `calendars.insert` im Kalender Backend gespeichert,

```
calendars.insert(new Array(entry));
```

oder mit `swing.openCalendarEntry` zum Bearbeiten geöffnet.

```
swing.openCalendarEntry([entry], null, true, null);
```

Der zweite Parameter gibt dabei an, welches Element bei Serienterminen geöffnet werden soll. Der dritte Parameter gibt an, ob der Kalendereintrag im aktiven oder in einem neuen Fenster geöffnet werden soll. Der letzte Parameter gibt die Übergabeparameter für den Kalendereintrag an.

2.3. Pflichtangaben

Um einen Kalendereintrag anlegen zu können, müssen mindestens folgende Keys in der Map angegeben werden.

Für Aufgaben und Termine:

Feld	Beschreibung
STATUS	Statuskennung des Eintrags (s. Statusliste unten).
TYPE	Der Typ des Kalendereintrags (`calendars.VEVENT` oder `calendars.VTODO`).
USER	Der Eigentümer des Kalendereintrags.

Für Termine zusätzlich:

Feld	Beschreibung
DTSTART	Start-Datum des Termins.
DTEND / DURATION	End-Datum des Termins oder Dauer.

2.4. Angabe von Benutzern

Für einen Kalendereintrag werden generell der Ersteller eines Eintrags und die betroffenen Benutzer unterschieden.

In jedem Fall werden die Benutzer als Kalender-User angegeben. Diese können mit der Methode `getCalendarUser` (o.ä.) ermittelt werden.

```
entry[calendars.USER] =
calendars.getCalendarUser(vars.getString("$sys.user"));
```

Die betroffenen Benutzer sind die Teilnehmer eines Termins oder die Zuständigen für eine Aufgabe. Für jeden einzelnen betroffenen Benutzer muss der Kalender-User ermittelt werden. Für das Feld `AFFECTEDUSERS` wird ein Array mit Kalender-Usern mit der Methode `encodeMS` multistring-codiert und dem Feld zugewiesen.

```
var users = new Array();
users[0] = calendars.getCalendarUser("Lola Locker");
users[1] = calendars.getCalendarUser("Günter Moser");
entry[calendars.AFFECTEDUSERS] = text.encodeMS(users);
```

2.5. Angabe des Status

Je nach Typ des Kalendereintrags können unterschiedliche Statuswerte vergeben werden.

Für Aufgaben:

STATUS	Beschreibung
NEEDSACTION	Noch nicht begonnen
INPROCESS	In Arbeit
COMPLETED	Erledigt

Für Termine:

STATUS	Beschreibung
TENTATIVE	Angefragt
CONFIRMED	Bestätigt
CANCELLED	Abgesagt

Abhängig vom verwendeten Kalenderbackend kann auch `calendars.STATUS_BUSY` der Rückgabewert für bestätigte Termine sein.

2.6. Angabe von Zeitraum und Fälligkeit

Sowohl Aufgaben als auch Termine besitzen ein Start-Datum. Das Start-Datum wird mit dem Feld `DTSTART` angegeben.

```
entry[calendars.DTSTART] = vars.getString("$sys.date");
```

Ein Termin besitzt ein End-Datum (`DTEND`),

```
entry[calendars.DTEND] = eMath.addInt(vars.getString("$sys.date"),
datetime.ONE_DAY);
```

eine Aufgabe eine Fälligkeit (`DUE`):

```
entry[calendars.DUE] = eMath.addInt(vars.getString("$sys.date"),
datetime.ONE_DAY);
```

Beide Werte können alternativ über die Dauer (`DURATION`) gesetzt werden, die zum Start-Datum hinzugezählt wird.

```
entry[calendars.DURATION] = String(datetime.ONE_DAY);
```

Diese Angaben erfolgen immer als Long-Werte. Diese müssen allerdings dem Kalenderobjekt als String übergeben werden.

Zum Berechnen von Zeiträumen leisten Ihnen die Methoden `datetime.toDate` und `datetime.toLong` mit geschickt gewählten Format-Mustern, sowie die Konstanten `ONE_SECOND`, `ONE_MINUTE`, `ONE_HOUR`, `ONE_DAY` und `ONE_WEEK` gute Dienste.

3. Lesen eines Kalendereintrags

Beim Lesen von Kalendereinträgen wird zwischen dem Lesen eines einzelnen Eintrags (einer Aufgabe oder eines Termins) und dem Lesen mehrerer Einträge, die bestimmte Suchbedingungen erfüllen, unterschieden.

Ein einzelner Kalendereintrag wird mit Hilfe der Methode `calendars.getEntry(pUID)` gelesen. Der Parameter ist die eindeutige ID für den Kalendereintrag. Da dazu aber diese eindeutige Kennung bereits vorliegen muss, werfen wir einen Blick darauf, wie sich diese ID in Erfahrung bringen lässt: die Suche nach Kalendereinträgen.

3.1. Suchen von Kalendereinträgen

Für die Suche nach Kalendereinträgen wird ein Array definiert, das die Suchkriterien enthält. Für die Suche nach Einträgen können die Eigenschaften aus der folgenden Liste verwendet werden.

- END Endedatum des Termins bzw. der Aufgabe
- START Startdatum des Termins bzw. der Aufgabe
- STATUS Status der Aufgabe
- TYPE Typ des Eintrags (`calendars.VEVENT` | `calendars.VTODO`)
- UID Die eindeutige Kennung des Kalendereintrags
- USER Der Eigentümer des Kalendereintrags

Es kann mehr als eine Suchbedingung für `calendars.getEntries()` verwendet werden. Daher muss bei der Erzeugung des Arrays auch die Anzahl der Bedingungen über den Schlüssel `COUNT` mit angegeben werden (s. Beispiel unten). Die definierten Bedingungen werden logisch UND-verknüpft.

Um die Eigenschaften den einzelnen Bedingungen zuzuordnen, muss jede Eigenschaft einen Postfix mit der Nummer der Bedingung erhalten. Für die Eigenschaft `STATUS` der ersten Bedingung wird also der Name `STATUS_1` erzeugt!

Das Codefragment unten sucht nach allen Aufgaben des Benutzers **Admin**, die den Status "In Bearbeitung" besitzen.

```
var condition = new Array();
condition["COUNT"] = "1";
condition["TYPE_1"] = calendars.VTODO;
condition["USER_1"] = "Admin";
condition["STATUS_1"] = calendars.STATUS_INPROCESS;
var entries = calendars.getEntries(condition);
```

Sie erhalten dann ein Array mit den Einträgen aus dem Kalender, die diese Bedingung erfüllen. Jedes Element ist ein Array, bei dem für Aufgaben nur das Element mit dem Index 0 verwendet wird. Die Erklärung, warum `calendars.getEntries()` ein Array liefert, finden Sie im Abschnitt über Serientermine.

3.2. Auslesen eines einzelnen Kalendereintrags

3.2.1. Zugriff direkt über ID

Über die Funktion `calendars.getEntry` kann ein einzelner Kalendereintrag ausgelesen werden.

Die ID eines Eintrags ist in der Regel in der folgenden Form aufgebaut:

```
; UID; User; reccurrenceID
```

Dieser Aufbau wird im ADITO-Basissystem in allen Kalender-Tabellen verwendet, ist also keine vom System streng vorgegebene Richtlinie.

- `UID`: Tatsächliche ID des Kalendereintrages
- `User`: ADITO-Benutzerlogin des Kalendereintrages
- `ReccurrenceID`: ID der Serientermin-Wiederholung, wenn „null“ übergeben wird, dann wird das erste Element verwendet.

Das heißt, die ID befindet sich an erster Stelle ([0]) des Multistrings. Wollen Sie also mit Hilfe dieser IDs einen Kalendereintrag auslesen, kann das mit dem folgenden Befehl erfolgen:

```
var entryid = text.decodeMS(text.decodeFirst(
vars.getString("$comp.tbl_Termine")));


var entry = calendars.getEntry(entryid[0], null,
entryid[1],calendars.VEVENT);
```

3.2.2. Einträge über ID suchen

Verwenden Sie die Suchformel aus dem vorherigen Kapitel, dann können Sie bei `getEntry` natürlich direkt auf diese ID zugreifen.

Der Rückgabewert von `getEntry` ist stets ein mehrdimensionales Array, deswegen muss selbst bei Angabe einer expliziten ID immer noch auf das erste Element des Eintrags zugegriffen werden.

Danach können Sie auf alle Elemente des Kalendereintrages zugreifen wie unter Kapitel 2 beschrieben, z.B.

```
question.showMessage(entry[calendars.SUMMARY]);
```

4. Verknüpfungen

Kalendereinträge können in ADITO mit Datensätzen anderer Tabellen im Datenmodell verknüpft werden.

4.1. Verknüpfungen anlegen

Verknüpfungen werden beim Anlegen ähnlich formuliert wie die Bedingungen (Conditions) zum Suchen von Kalendereinträgen.

Mit der Eigenschaft `calendars.LINKS` eines Kalendereintrags wird zunächst die Anzahl Verknüpfungen angegeben. Für jede Verknüpfung müssen dann folgende Informationen angegeben werden. Dabei werden hier die Schlüssel als Zeichenketten (Strings) und nicht als JDito-Konstanten angegeben. In jedem Schlüssel wird durch ein angehängtes "_<Nummer>" die Nummer der Verknüpfung angegeben, zu der die Information gehört.

SCHLÜSSEL	Beschreibung
LINK_ALIAS	Der DB-Alias, in dem der zu verknüpfende Datensatz liegt.
LINK_TABLE	Die Tabelle, in der der zu verknüpfende Datensatz liegt.
LINK_IDCOLUMN	Die Primärschlüsselspalte der Tabelle.
LINK_DBID	Die ID des Datensatzes, der verknüpft werden soll.
LINK_FRAME	Der Frame, in dem der verknüpfte Datensatz geöffnet werden soll.
LINK_TITLE	Titel für die Verknüpfung.

Folgendes Codebeispiel zeigt den Aufbau einer solchen Verknüpfung:

```
entry[calendars.LINKS] = "1";
entry["LINK_ALIAS_1"] = "AO_DATEN";
entry["LINK_TABLE_1"] = "ORG";
entry["LINK_IDCOLUMN_1"] = "ORGID";
entry["LINK_DBID_1"] = "1022";
entry["LINK_FRAME_1"] = "ORG";
entry["LINKE_TITLE_1"] = "Firma";
```

4.2. Arbeiten mit Verknüpfungen

Im folgenden Beispiel wird davon ausgegangen, dass in der Komponente `$comp.tblEvents` Kalender-IDs zurückgegeben werden.

```
var uid = text.decodeFirst(vars.getString("$comp.tblEvents"));

var entry = calendar.getEntry(uid);

question.showMessage("LINKS = " + entry[calendars.LINKS]);
question.showMessage(entry["LINK_ALIAS_1"]);
question.showMessage(entry["LINK_TABLE_1"]);
question.showMessage(entry["LINK_FRAME_1"]);
```

5. Erinnerungen

Bei Kalendereinträgen können Sie sich zu einem bestimmten Zeitpunkt oder in einem bestimmten Zeitrahmen vor Fälligkeit erinnern lassen. Die Erinnerung wird im ADITO-Client mittels blinkendem Symbol und Notifikation im Windows System-Tray oder als Hinweiszahl im Mac OS X-Dock angezeigt.

Um einen Reminder hinzuzufügen, muss der `HASREMINDER`-Wert auf `true` gesetzt werden.

```
entry[calendars.HASREMINDER] = "true";
```

5.1. Typen von Erinnerungen

5.1.1. Festes Datum

Erinnerungen mit festem Datum erscheinen zum festgelegten Zeitpunkt. Zuvor muss der Reminder vom Typ absolut gesetzt sein. Diese werden vom Benutzer festgelegt.

```
var absolut = event[calendars.REMINDER_ABSOLUT];
if(absolut == "true")
{
var reminder_date = vars.getString("$sys.date");
entry[calendars.REMINDER_DATE] = reminder_date;
}
```

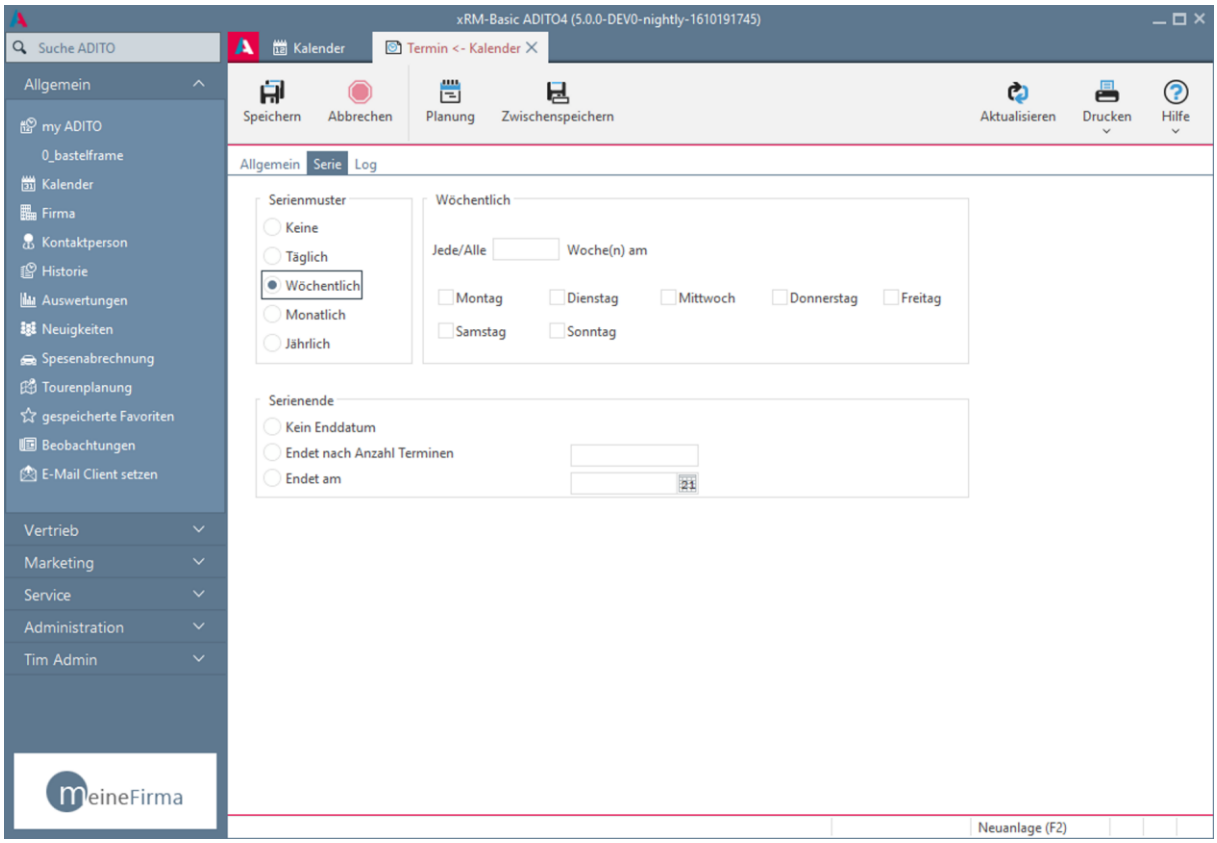
5.1.2. Zeit vor Fälligkeit

Erinnerungen vom Typ Duration erscheinen in einer gewissen Zeitspanne vor Fälligkeit des Termins / der Aufgabe.

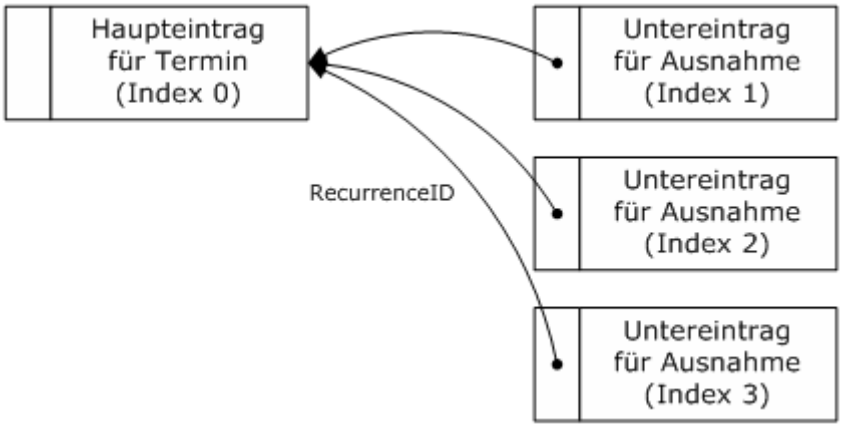
```
var reminder_duration = vars.getString("$comp.reminder_duration");
entry[calendars.REMINDER_DURATION] = reminder_duration;
```

6. Serientermine

Bei Terminen wird zwischen Einzelterminen und Terminserien unterschieden. Ein Einzeltermin ist genau dies: ein Termin ohne Wiederholungen oder Ausnahmen. Eine Serie besteht aus sich wiederholenden Terminen. Das Wiederholungsmuster für die Serie kann im Kalender von ADITO online festgelegt werden.



Innerhalb einer solchen Terminserie kann es Ausnahmen geben, einzelne Elemente der Serie können gelöscht werden oder an einem anderen Ort stattfinden. Ein Element reicht also für die Wiedergabe dieser Informationen nicht aus. Daher wird beim Lesen von Kalendereinträgen für einen Eintrag immer ein Array zurückgeliefert. Für einen Einzeltermin oder eine Aufgabe ist nur das erste Element in diesem Array (das mit dem Index 0) belegt. Bei einer Terminserie kommt für jede Ausnahme ein Element hinzu. Dies zeigt die folgende Grafik.



6.1. Feststellen, ob ein Termin ein Serientermin ist

Serientermine besitzen mindestens eine der drei Eigenschaften `RDATE`, `RRULE` oder `EXDATE`. Um festzustellen, ob es sich bei einem Kalendereintrag um einen Serientermin handelt, müssen Sie diese drei Eigenschaften auf einen Wert abfragen. Ist keine der Eigenschaften gesetzt, handelt es sich um einen Einzeltermin. Das Listing unten zeigt dies am Beispiel eines Kalendereintrags, der in der Variablen `entry` gespeichert ist.

```
var elem = entry[i];


if( (elem[0][calendars.RDATE] != undefined)
    || (elem[0][calendars.RRULE] != undefined)
    || (elem[0][calendars.RECURRENCEID] != undefined))
 {
  question.showMessage("Serientermin: " +
elem[0][calendars.SUMMARY]);
 }
else
 {
  question.showMessage("Einzeltermin: " +
elem[0][calendars.SUMMARY]);
 }
```

6.2. Haupteintrag und Untereintrag bei einem Serientermin unterscheiden

Ein Serientermin besitzt mindestens einen Eintrag - den Haupteintrag. Dieser hat, wie im obigen Beispiel gezeigt, eine der drei Erkennungseigenschaften für eine Terminserie gesetzt. Wie lassen sich nun Haupteintrag und Untereinträge für Ausnahmen unterscheiden? Die Eigenschaft `RECURRENCEID` ist die Kennung für einen Untereintrag. Wenn also einer der beiden Eigenschaften `RDATE` oder `RRULE` gesetzt sind, handelt es sich um den Haupteintrag, bei `RECURRENCEID` um den Untereintrag einer Terminserie. Dazu erweitern wir das Beispiel von oben um die Unterscheidung der beiden Eintragstypen.

```
var elem = entry[i];


if( (elem[0][calendars.RDATE] != undefined)
    || (elem[0][calendars.RRULE] != undefined)
    || (elem[0][calendars.RECURRENCEID] != undefined))
 {
  for(var j = 0; j < entries[i].length; j++)
  {
   var item = elem[j];
  if( (item[calendars.RDATE] != undefined)
        || (item[calendars.RRULE] != undefined))
  {
   question.showMessage("Serie Haupteintrag " +
elem[0][calendars.SUMMARY]);
  } else
  {
   question.showMessage("Serie Untereintrag " +
elem[0][calendars.SUMMARY]);
  }
  }
 }
 else
 {
  question.showMessage("Einzeltermin: " +
elem[0][calendars.SUMMARY]);
 }
```

6.3. Gelöschte Elemente einer Serie finden

Wenn aus einer Terminserie einzelne Elemente gelöscht wurden, finden Sie diese in der Eigenschaft `EXDATE` des Haupteintrags.

6.4. Aufbereitete Termine lesen

Zusätzlich zur Methode `getEntries()` existiert in JDito noch eine zweite Methode mit der Signatur `getExpandedEntries(pCondition, pStart, pEnd)`. Diese liefert Terminserien aufbereitet zurück. Das bedeutet, dass Ausnahmetermine und gelöschte Serienelemente gleich berechnet werden und das Ergebnis dieser Methode direkt für die Anzeige verwendet werden kann. Bei der Verwendung von `getEntries()` müssten Sie sonst aus der Definition einer Terminserie die tatsächlichen Instanzen erst berechnen. Da das Berechnen der tatsächlich stattfindenden Termine über einen längeren Zeitraum einen

erheblichen Aufwand bedeutet, muss bei `getExpandedEntries()` zusätzlich Beginn und Ende des Zeitintervalls angegeben werden.

Das folgende Listing liefert alle Termine des angemeldeten Benutzers im Zeitraum vom 01.05.2007 bis zum 31.05.2007 als tatsächliche Termineinträge, also mit expandierten Terminserien.

```
var dtStart = datetime.toLong("2007-05-01", "yyyy-MM-dd",
"Europe/Berlin");
var dtEnd = datetime.toLong("2007-05-31", "yyyy-MM-dd",
"Europe/Berlin");
var condition = new Array();
condition["COUNT"] = "1";
condition["TYPE_1"] = calendars.VEVENT;
condition["START_1"] = String(dtStart);
condition["END_1"] = String(dtEnd);
condition["USER_1"] = vars.getString("$sys.user");
var entries = calendars.getExpandedEntries(condition, dtStart,
dtEnd);
```

Da bei `getExpandedEntries()` alle Ausnahmen und Löschungen aufgelöst werden, ist für jedes Element in `entries` auch nur der Index 0 gefüllt (`entries[i][0]` verwenden). Damit sollten Sie für die Anzeige der Termine eines bestimmten Zeitraums immer die Methode `getExpandedEntries()` verwenden.

6.5. Terminserien anlegen

Um Terminserien anzulegen verwenden Sie den Schlüssel `calendars.RRULE` eines Kalender-Objekts. Als Wert wird ein Array mit einem String-Element übergeben. Dieser String wird wiederum aus Schlüssel-Wert-Paaren zusammengesetzt, die mit Strichpunkten (";") voneinander getrennt werden.

Mögliche Schlüssel sind:

Schlüssel	Beschreibung
FREQ	Frequenz, Wiederholung (DAILY, WEEKLY, MONTHLY, YEARLY).
INTERVAL	Intervall, Abstand zwischen zwei Serienelementen.
BYDAY	Wochentag, mehrere werden durch Komma getrennt (MO, TU, WE, TH, FR, SA, SU). Eine vorangestellte ganze Zahl signalisiert den x-ten, also 1MO den ersten Montag im jeweiligen Zeitraum. Negative Zahlen signalisieren ein

	Zählen vom Ende her, also -1MO den letzten Montag im jeweiligen Zeitraum.
BYMONTHDAY	Tag im Monat, Angabe als ganze Zahl zwischen 1 und 31, mehrere werden durch Komma getrennt. -1 bedeutet "letzter" (d.h. 31, 30 oder 28), -2 bedeutet "vorletzter" (d.h. 30, 29 oder 27).
BYMONTH	Monat, Angabe als ganze Zahl zwischen 1 und 12, mehrere werden durch Komma getrennt.
WKST	Bei FREQ=WEEKLY, Angabe des ersten Tags der Woche, also SU oder MO.
COUNT	Serienende nach Anzahl Terminen.
UNTIL	Serienende mit einem bestimmten Datum, dieses Datum wird im Format "yyyyMMdd\'T\'HHmmss\'Z\'" angegeben.

Beispiele:

1. Alle zwei Tage

```
entry[calendars.RRULE] = new Array("FREQ=DAILY;INTERVAL=2");
```

2. Jeden zweiten Montag

```
entry[calendars.RRULE] = new
Array("FREQ=WEEKLY;INTERVAL=2;BYDAY=MO");
```

3. Alle drei Monate am 5. des Monats

```
entry[calendars.RRULE] = new
Array("FREQ=MONTHLY;INTERVAL=3;BYMONTHDAY=5");
```

4. Alle drei Monate am letzten des Monats

```
entry[calendars.RRULE] = new
Array("FREQ=MONTHLY;INTERVAL=3;BYMONTHDAY=-1");
```

5. Jeden ersten Montag jeden vierten Monat

```
entry[calendars.RRULE] = new
Array("FREQ=MONTHLY;INTERVAL=4;BYDAY=1MO");
```

6. Jedes Jahr am 2.6.

```
entry[calendars.RRULE] = new
Array("FREQ=YEARLY;BYMONTHDAY=2;BYMONTH=6");
```

7. Jeden ersten Freitag im August

```
entry[calendars.RRULE] = new
Array("FREQ=YEARLY;BYDAY=1FR;BYMONTH=8");
```

8. Alle zwei Tage, endet nach 10 Terminen

```
entry[calendars.RRULE] = new
Array("FREQ=DAILY;INTERVAL=2;COUNT=10");
```

9. Alle zwei Tage, endet am 31.12.2007

```
var until = datetime.toDate(datetime.toLong("31.12.2007",
"dd.MM.yyyy"), "yyyyMMdd\'T\'HHmmss\'Z\'", "UTC");

entry[calendars.RRULE] = new
Array("FREQ=DAILY;INTERVAL=2;UNTIL" + until);
```

Viele Beispiele zu Wiederholungen finden Sie in RFC 2445 ("Internet Calendaring and Scheduling Core Object Specification (iCalendar)").

6.6. Terminserien lesen

Mit der Methode `getRuleAsMap()` kann die `RRULE`-Eigenschaft eines Kalenderelements als JavaScript-Map ermittelt werden. Auf die einzelnen Elemente der RRULE kann dann über den Elementnamen als Key zugegriffen werden.

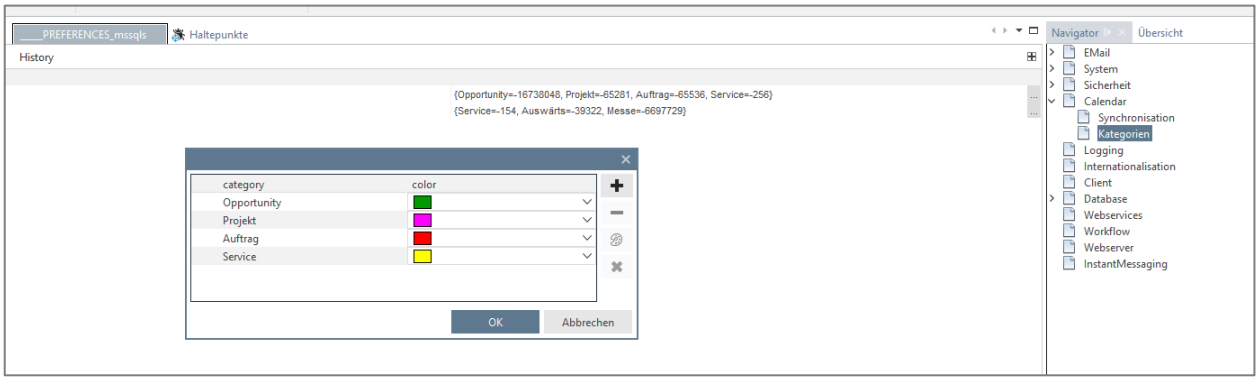
```
var condition = new Array();
condition["COUNT"] = "1";
condition["TYPE_1"] = calendars.VEVENT;
var entries = calendars.getEntries(condition);

for (var i=0; i<entries.length; i++)
{
 var entry = entries[i][0];
 var rrule = entry[calendars.RRULE][0];
 var rrulemap = calendars.getRuleAsMap(rrule);
 var freq = rrulemap["FREQ"];
}
```

7. Verwenden von Kategorien

7.1. Kategorien definieren

Die Definition von Kategorien erfolgt im Designer mit dem Preferences-Editor [PS]. Wechseln Sie dort in den Reiter `Calendar`. Dort können Sie in der Gruppe "Kategorien" sowohl für Termine als auch für Aufgaben die Liste der Kategorien definieren. Änderungen an dieser Einstellungen werden erst nach einem Neustart des Servers wirksam.



7.2. Auslesen der Kategorien eines Eintrags

Um die für einen Kalendereintrag definierten Kategorien auszulesen, verwenden Sie die Eigenschaft `CATEGORIES` des jeweiligen Eintragselements. Diese liegen als Multi-String codiert vor. Zur Umwandlung in ein Array verwenden Sie die Methode `decodeMS()`. Im Beispiel unten wird ein Kalendereintrag direkt über die Methode `getEntry()` gelesen und anschließend wird die Liste der Kategorien mit `decodeMS()` in ein Array gewandelt und in der Variablen `cat` gespeichert.

```
// Laden des Kalendereintrags
var entry = calendar.getEntry( uidFuerDenEintrag );
// Speichern der Kategorien in einem Array
var cat = text.decodeMS(entry[calendars.CATEGORIES]);
```

Wenn Sie die Kategorien in einer Listenkomponente auf einem Frame anzeigen wollen, können Sie den Inhalt der Eigenschaft `CATEGORIES` direkt zum Füllen der Komponente verwenden.

7.3. Setzen einer Kategorie

Um eine Kategorie in einem Kalendereintrag zu setzen, ist einfach der entsprechende Multistring zu erweitern. Soll beispielsweise immer eine Kategorie vom Typ "Drei" einem Eintrag hinzugefügt werden, kann das wie folgt geschehen:

```
   var cats = vars.get("$comp.categories");
    cats += "; Drei;"
    event[calendars.CATEGORIES] = cats;
```

8. Termine absagen

Einzelne Kalendertermine können in ADITO abgesagt werden. Dabei werden nicht die ganzen Termine abgesagt, vielmehr kann jeder Teilnehmer selbst den Status setzen. Hierfür gibt es die Eigenschaft `PARTSTAT`, die für jeden Teilnehmer einzeln gesetzt werden kann.

Für `PARTSTAT` gibt es folgende Werte:

- `NEEDS_ACTION`: Muss noch beantwortet werden.
- `ACCEPTED`: Kalenderelement angenommen.
- `DECLINED`: Kalenderelement abgelehnt.

8.1. Beispiel: Kalenderelement absagen.

8.1.1. Kalendereintrag anlegen

Im Folgenden ein Codebeispiel, welches einen Kalendereintrag anlegt:

```
// Anlegen eines einfachen Kalenderelements
var entry = new Array();

entry[calendars.TYPE] = calendars.VEVENT;
entry[calendars.STATUS] = calendars.STATUS_CONFIRMED;
entry[calendars.USER] = calendars.getCalendarUser("Admin");
entry[calendars.DTSTART] = vars.getString("$sys.date");
entry[calendars.DTEND] =
String(parseInt(vars.getString("$sys.date")) +
parseInt(datetime.ONE_HOUR));
entry[calendars.SUMMARY] = "Testtermin " +
datetime.toDate(vars.getString("$sys.date"), "HH:mm");
entry[calendars.DESCRIPTION] = "Das ist der Text."

var users = new Array();
users[0] = calendars.getCalendarUser("Lola Locker");
entry[calendars.AFFECTEDUSERS] = text.encodeMS(users);

calendars.insert(new Array(entry));
```

In einem ADITO-Referenzsystem kann dieser Code ausgeführt werden (Das Referenzsystem erhalten Sie als Kunde mit Wartungsvertrag auf Wunsch von ADITO).

Der Kalendereintrag, der hier angelegt wird, wird für die Benutzer **Admin** und **Lola Locker** angelegt. Lola Locker wird als weiterer Benutzer (im Property `AFFECTEDUSERS`) angelegt.

Die ID des Kalendereintrags kann nun über eine SQL-Abfrage auf die Systemtabelle `ASYS_CALENDAR` ermittelt werden.

8.1.2. Kalenderelement absagen: ADITO-Kalender (Backend:DB)

Um für Lola Locker den Kalendereintrag abzusagen, kann folgender Code ausgeführt werden:

```
// Kalender-Eintrag auslesen und absagen (DB-Kalender)
var entryuid = "f6254678-ccf4-4197-b0a0-88a40232669f"; //ID ist vom oben angelegten Termin
var entry = calendars.getEntry(entryuid, null, "Admin");



var att = entry[calendars.ATTENDEES];


// Affectedusers: Absagen
var au = entry[calendars.AFFECTEDUSERS];
var auD = text.decodeMS(au); // Affected Users decodiert


var au1 = text.decodeMS(auD[1]); // Decodieren eines Users (Lola Locker)
var au1N = new Array(); // Neues Array, da au1 ein Array mit fester Länge ist


// Übernahme des alten au1 Array in das neue au1N-Array
for(var i = 0; i < au1.length; i++)
{
    au1N.push(au1[i]);
}


// Jetzt zusätzliche Absage des PARTSTAT-Arrays
au1N.push("PARTSTAT:DECLINED"); // Sagt ab, ABER! Bleibt in ASYS_CALENDAR_AU bestehen


// Affected-User wieder in alles Array übernehmen
auD[1] = text.encodeMS(au1N);
au = text.encodeMS( auD );
entry[calendars.AFFECTEDUSERS] = au;


calendars.updateEntry(entry);
```

(Hier wird die ID vorausgesetzt – wenn Sie den Code nachprüfen, müssen Sie den Wert der Variable entryuid entsprechend ändern!)

`PARTSTAT:DECLINED` muss als Wert in das Array des abzusagenden Users für `AFFECTEDUSERS` eingetragen werden. Dazu ist es notwendig, ein neues Array anzulegen, da das decodierte User-Array eine feste Größe hat und bspw. nicht über `push()` erweitert werden kann.

8.1.3. Kalenderelement absagen: Anbindung Exchange Webservices

Das Element `PARTSTAT` kann nicht direkt an das Exchange-Kalenderbackend gesetzt werden, da dieses Element von Exchange aus „nur lesen“ ist. Stattdessen gibt es den Status des Kalendertermins, der für den jeweils angemeldeten Benutzer gesetzt werden muss. Das wird im folgenden Codebeispiel gemacht: Der Kalendereintrag wird für den aktuell angemeldeten User abgesagt:

```
var entryuid =
"040000008200E00074C5B7101A82E008000000004B647DF04332CF0100000000000
0000010000000C6BFC0EFC35C804F863476C4A9D2D76B";

// ID ist vom oben angelegten Termin, ausgelesen über die
ASYS_UIDRESOLVER

var entry = calendars.getEntry(entryuid, null, "Admin");


entry["X-ADITO-STATUSACTION"] = "DECLINE";


calendars.updateEntry( entry );
```

Die ID wurde diesmal über die Systemtabelle `ASYS_UIDRESOLVER` ermittelt. Das Element, das für den aktuellen User den Eintrag abfragt, ist `X-ADITO-STATUSACTION`. Mögliche Werte sind:

- ACCEPT
- DECLINE