Series25®

Understanding vCalendar

25Live Pro

CollegeNET®

CollegeNET, Inc.
805 SW Broadway, Suite 1600
Portland, Oregon 97205
(503) 973-5200
corp.collegenet.com

Series25-SIS Interface version 3.4

Introduction

About this manual

Purpose of this manual

The purpose of this manual is to help you understand vCalendar, the data format used by the CollegeNET Series25-SIS Interface to represent the class data that is shared between your student information system (SIS) and 25Live.

Who should use this manual

This manual is a companion document to the *Implementing and Using the Series25-SIS Interface Version 3.4* manual. It is written for technical members of your Series25 implementation team and other interested parties who want to understand more about CollegeNET's implementation of the vCalendar data standard.

Series25 documentation, training, and support resources

All Series25® documentation, training, and support resources are accessible online from the Series25 Customer Resources site (login required):

http://knowledge25.collegenet.com/display/CustomerResources/Welcome+to+Series25+Customer+Resources

The support services available to you and how to obtain them are described in the *Series25 Customer Handbook* available here:

http://knowledge25.collegenet.com/display/CustomerResources/Contacting+Series25+Support

To get login credentials to the Customer Resources site, contact *support@collegenet.com*.

If you have comments

Please let us know how we're doing. Send an email comment to doc@collegenet.com if you:

- Discover an error in this document
- Find a concept or instruction confusing
- Have a suggestion to improve the document

2 Introduction to vCalendar

Interface communication between your SIS and 25Live is accomplished using a data exchange format standard called *vCalendar*.

What is vCalendar?

The vCalendar standard

vCalendar is a global format standard for the exchange of calendar and scheduling data that is platform and device independent. It was developed by a consortium of hardware and software manufacturers to enable "diverse communication and computing devices, applications, and services from competing vendors to interoperate in all environments."

vCalendar uses:

- Plain-text format (.vcs) ASCII files containing vCalendar and vEvent "objects"
- Object properties and values to represent calendar and meeting data

Why CollegeNET adopted the vCalendar standard

CollegeNET adopted the vCalendar standard for SIS-25Live class data integration for the following reasons:

- Using the vCalendar standard allows 25Live to process more class information from the SIS.
- vCalendar has been adopted and is widely used by major hardware and software companies to allow their devices and applications to communicate calendar data.
- Unique IDs and all data related to a class can be embedded in the vCalendar file, rather than having to create them in 25Live.

Note The CollegeNET vCalendar implementation has been tailored for academic scheduling and contains extensions to support specific Interface and 25Live functionality.

Location of full vCalendar specification

The full text of the vCalendar specification and other information on the vCalendar standard are available at:

- Internet Mail Corp: *www.imc.org/pdi/*
- IETF Scheduling Working Group: *www.imc.org/ietf-calendar/*

vCalendar objects

vCalendar files contain a vCalendar object which, in turn, contains one or more vEvent objects. See the sample vCalendar file below.

vCalendar objects

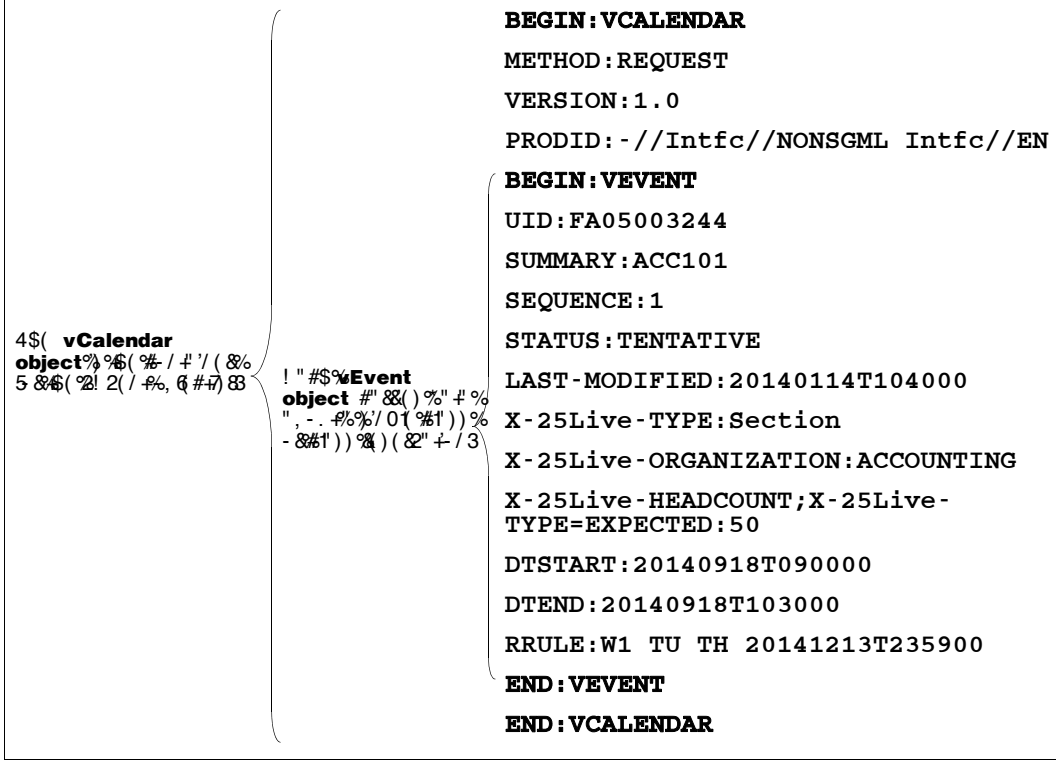
A vCalendar object:

- Specifies the method that should be used for processing the vEvent objects it contains (see "vCalendar methods" beginning on page 7)
- Identifies the vCalendar file and version

vEvent objects

A vEvent object defines a single class or class reservation.

Sample vCalendar file



vEvent object data

vEvent objects can include any of the following data elements to define a class, allowing you to share this data between your SIS and 25Live:

- Name, title, and description
- Unique identifier
- Event type
- Date range
- Reservation name(s)
- Reservation meeting pattern(s)
- Assigned room(s), including whether the assigned room can be "shared" with another unrelated class
- Related classes, such as those that are cross-listed, including the primary class of the related class group
- Version number
- Status
- Expected and registered head count
- Sponsoring department
- Room, campus partition, and room feature preferences
- Instructor

"Alien" events and "native" events

25Live regards the vEvent objects it imports from your SIS as "alien" events. This means that the vCalendar data in these objects—the data that is shared between your SIS and 25Live—is subject to special vCalendar processing rules that you set in the 25Live Administration Utility. (See the *Implementing and Using the Series25-SIS Interface Version 3.4* manual for more information on these processing rules.)

25Live regards events created in 25Live as "native" events. The data in these events is not subject to vCalendar processing rules.

vCalendar methods

Each vCalendar object has an identified method that defines the purpose of the object and how it should be processed. The method specified depends on which system generated the object—the Interface (from class data exported from your SIS) or 25Live.

Interface-generated methods

A vCalendar object generated by the Interface has either a *request* or a *cancel* method.

Request

A vCalendar object with a request method specifies that the object contains new classes that should be added to the Series25 database and/or changes to existing classes that should be made to the corresponding classes in the Series25 database.

Cancel

A vCalendar object with a cancel method specifies that the object contains classes that should be cancelled in the Series25 database.

25Live-generated methods

A vCalendar object generated by 25Live has either a *reply* or a *counter* method.

Reply

A vCalendar object with a reply method confirms that new and/or changed classes have been successfully added/modified in the Series25 database or that cancelled classes have been successfully cancelled in the Series25 database.

Reply objects are generated by 25Live in response to request objects. *No new data is transmitted in a reply.* 25Live uses the reply to let the SIS know that its database is synchronized with the SIS database.

Counter

A vCalendar object with a counter method specifies that the object contains new room assignments for class reservations where the original request didn't have a room assignment or where the revised head count of the reservation exceeded the capacity of the room(s) that had been previously assigned.

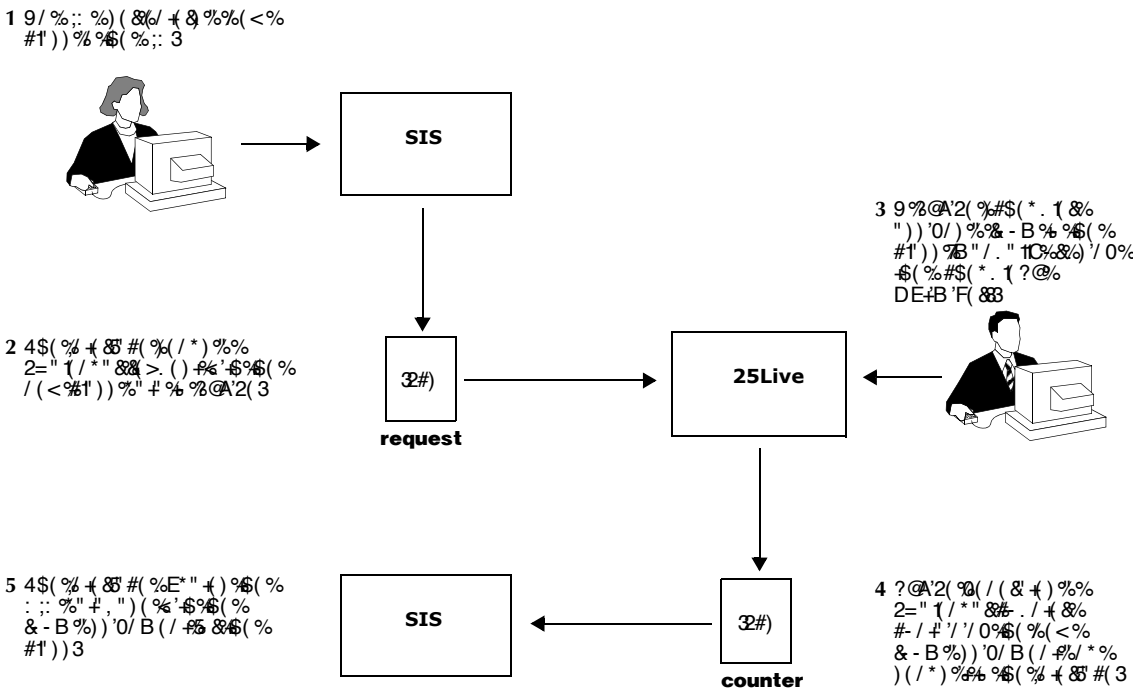
Note The Interface doesn't read the method of reply and counter files, relying instead on the Interface subdirectory location of these files to determine how to process them.

Data exchange examples

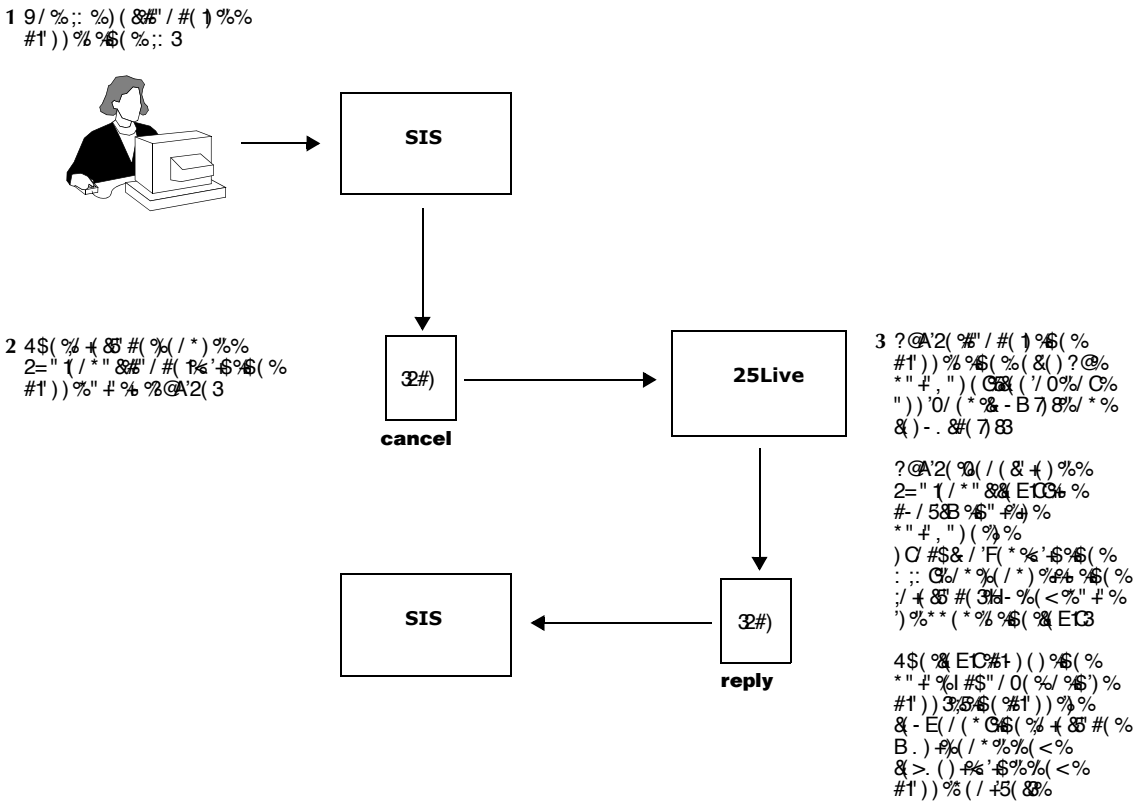
The examples below and on the next page illustrate the data exchange process.

These processes simulate a negotiation between two parties.

vCalendar Request-Counter Exchange



vCalendar Cancel-Reply Exchange



vCalendar properties

Description

vCalendar properties (data elements) and their associated values specify information about vCalendar and vEvent objects. For example, the METHOD vCalendar object property and its value identify the purpose of the object and how it should be processed, as discussed above. The LOCATION vEvent object property and its value specify the room assigned to the class reservation.

For a complete description of the supported properties, see "vCalendar Properties" beginning on page 22.

Extended (custom) properties

The vCalendar standard allows for vendor-specific properties. These "extended" (custom) properties begin with an "X-" followed by vendor identification. CollegeNET extended properties look like this:

```
X-R25-<property>
```

For example: `X-R25-TITLE`

Property parameters

Properties can be further defined by property parameters. In the CollegeNET implementation, this includes adding an X-25Live-TYPE parameter to further define a property value. In this example, a property parameter is used to specify the registered (rather than expected) head count of a class:

```
X-R25-HEADCOUNT;X-25Live-TYPE=REGISTERED:47
```

The X-R25-ID parameter

25Live assigns a unique internal identifier to every data element in its database. These identifiers allow the Interface to uniquely identify and respond to each 25Live data element.

For example, the 25Live "Section" event type might be assigned the value "5" in the Series25 database tables:

```
X-25Live-TYPE;X-25Live-ID=5:Section
```

Object delimiter properties

All vCalendar objects are delimited by these BEGIN and END properties:

```
BEGIN:VCALENDAR
...
END:VCALENDAR
```

All vEvent objects are delimited by these BEGIN and END properties:

```
BEGIN:VEVENT
...
END:VEVENT
```

Required vCalendar object properties

In addition to the BEGIN and END delimiter parameters, all vCalendar objects contain the following properties:

METHOD: The purpose of the object and how it should be processed. See "vCalendar methods" on page 7.

PRODID: The product that created the vCalendar object—the Interface or 25Live

VERSION: The supported vCalendar version—currently 1.0

Required vEvent object properties

In addition to the BEGIN and END delimiter properties, all vEvent objects contain the following properties, regardless of method:

SEQUENCE: The revision level of the vEvent object. See "The SEQUENCE property" beginning on page 13.

UID: The vEvent object's persistent, globally-unique identifier—used to identify the class or class reservation

Example

The following vCalendar request file contains one class. The required vCalendar and vEvent object properties are in bold. In addition to the properties listed above that are required in every vCalendar file, vEvent objects in request files must also include SUMMARY, PRIORITY, STATUS, DTSTART, and DTEND properties and values.

```
BEGIN:VCALENDAR
METHOD:REQUEST
VERSION:1.0
PRODID:-//Intrfc//NONSGML Intrfc//EN
BEGIN:VEVENT
UID:FA05003244
SUMMARY:ACC101
SEQUENCE:1
PRIORITY:0
ATTENDEE;ROLE=INSTRUCTOR:jdoe@myu.edu
STATUS:TENTATIVE
LAST-MODIFIED:20140114T104000
X-R25-TYPE:Section
X-R25-ORGANIZATION:ACCOUNTING
X-R25-HEADCOUNT;X-R25-TYPE=EXPECTED:50
DTSTART:20140918T090000
DTEND:20140918T103000
RRULE:W1 TU TH 20141213T235900
END:VEVENT
END:VCALENDAR
```

Property syntax

Property:value syntax

A property and its value are separated by a colon without a space.

```
<property>:<value>
```

Example:

```
LOCATION:BCC305
```

Property;parameter syntax

A property and a parameter are separated by a semicolon without a space.

```
<property>;<parameter=value>:<value>
```

Example:

```
X-R25-HEADCOUNT;X-R25-TYPE=EXPECTED:30
```

Date/time syntax

Date/time values are expressed following the ISO 8601 standard (T = time):

```
<year><month><day>T<hour><minute><second>
```

Example:

```
20140912T14000000
```

Recurrence rule syntax

Values to express repeating occurrences of a class reservation are defined in the RRULE (recurrence rule) property.

Examples:

A class meets every week on Monday, Wednesday, and Friday for 10 weeks:

```
RRULE:W1 MO WE FR #10
```

W = frequency
1 = interval
MO, WE, FR = frequency modifiers or days of week
#10 = duration

A lab meets every other week on Tuesdays until December 18, 2014:

```
RRULE:W2 TU 20141218T000000
```

Multiple lines

If a property and its value require more than one line, the second line (and each successive line, if applicable) begins with a space.

The SEQUENCE property

The SEQUENCE property identifies the revision level of a vEvent object. By comparing the sequence value of the imported object to the version number of the class in the Series25 database, 25Live is able to determine if the shared class data has been modified in the SIS. If the value is higher, the shared data has been modified.

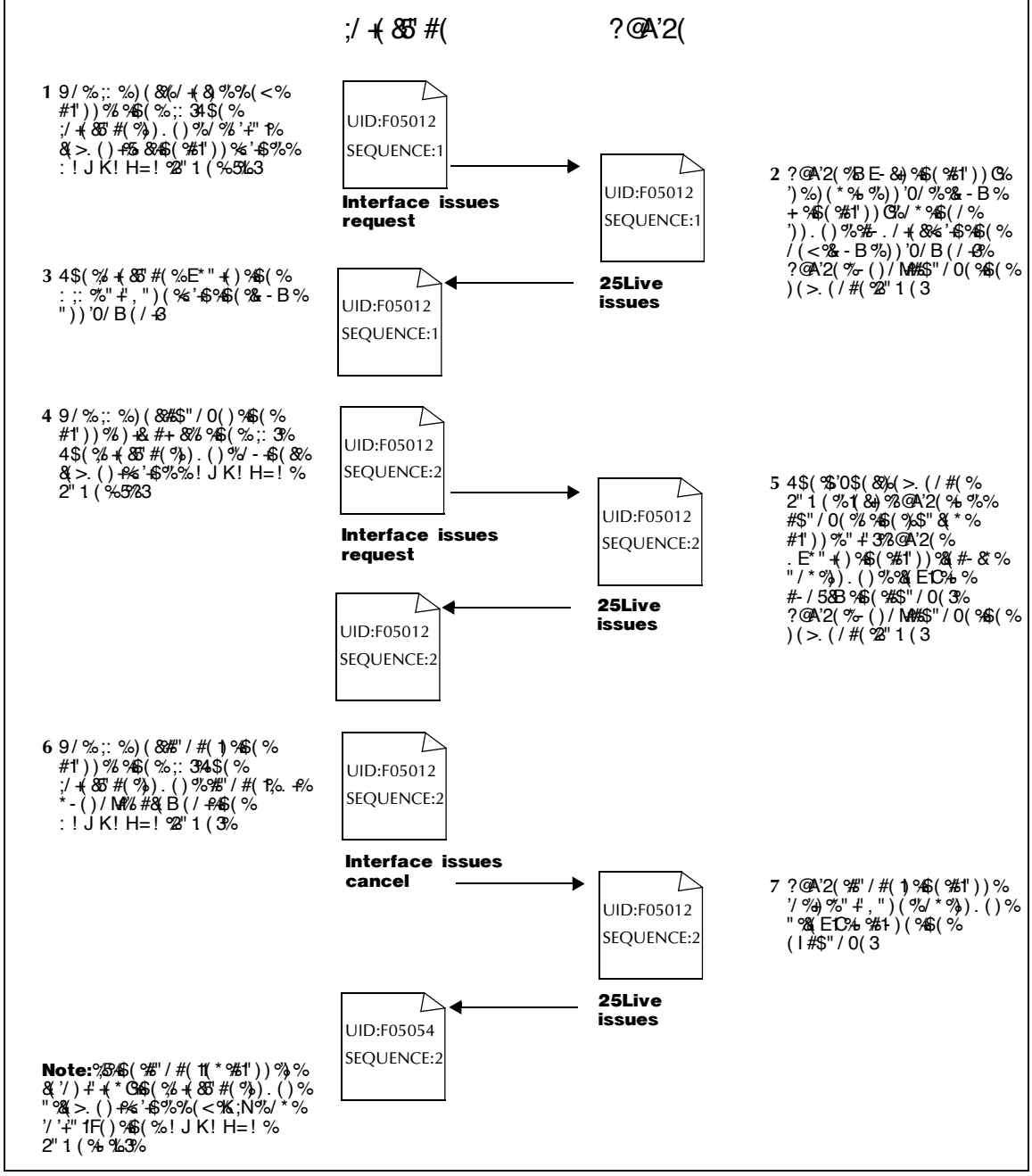
The Interface initializes the sequence value to 1 (one) in the vEvent object of each new class exported from your SIS, and then increments the value by 1 each time the class is modified in your SIS database and exported again.

Because your SIS "owns" the shared data, and because class additions and changes in the SIS initiate the data exchange via the Interface, only the Interface increments the SEQUENCE property value, never 25Live.

If a class is cancelled in your SIS, the Interface doesn't increment the sequence value in the vCalendar cancel file it creates. It is the CANCEL method in the file that directs 25Live to cancel the equivalent class record(s) in the Series25 database.

See the data flow example on page 14.

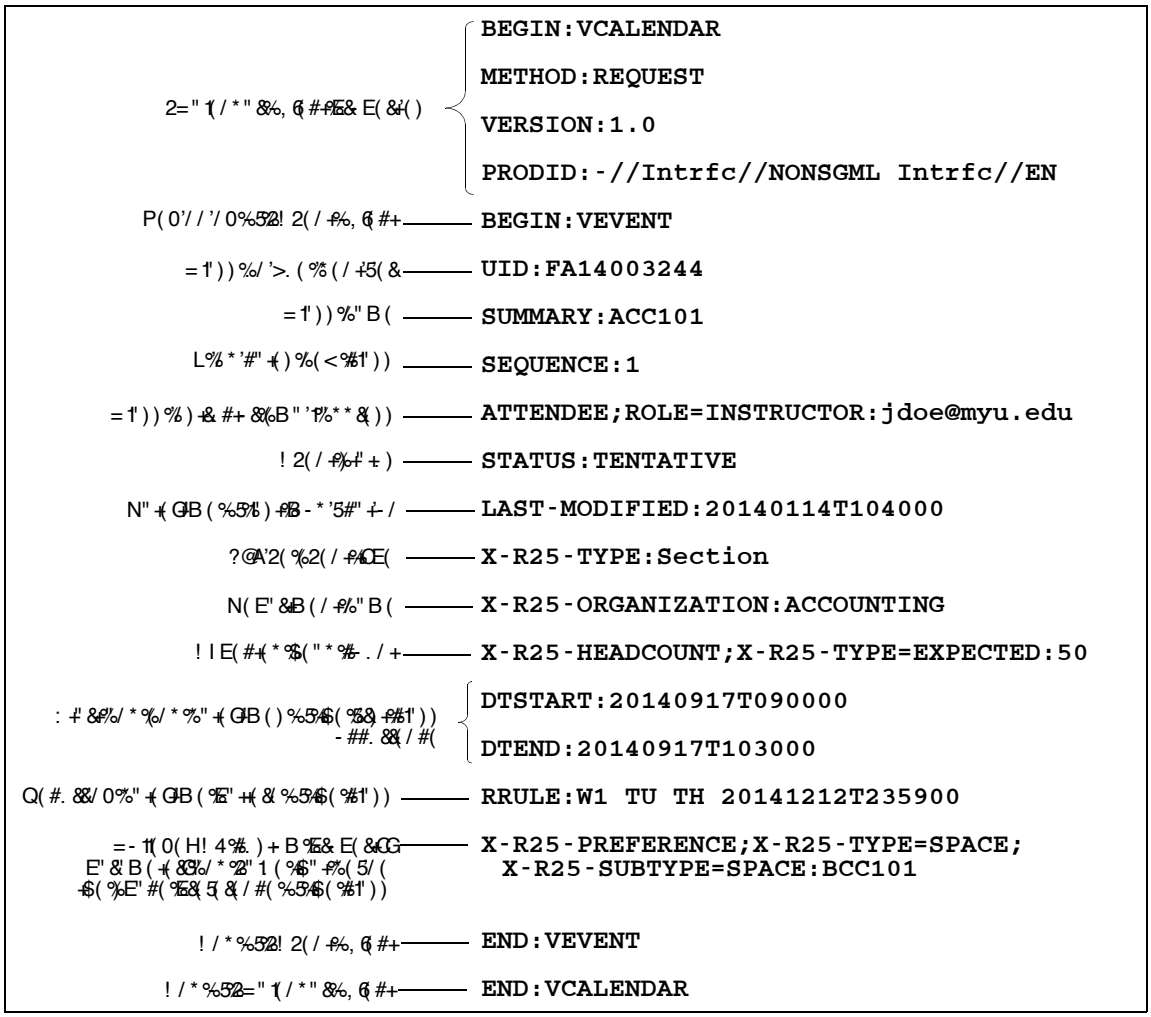
vCalendar SEQUENCE property data flow example



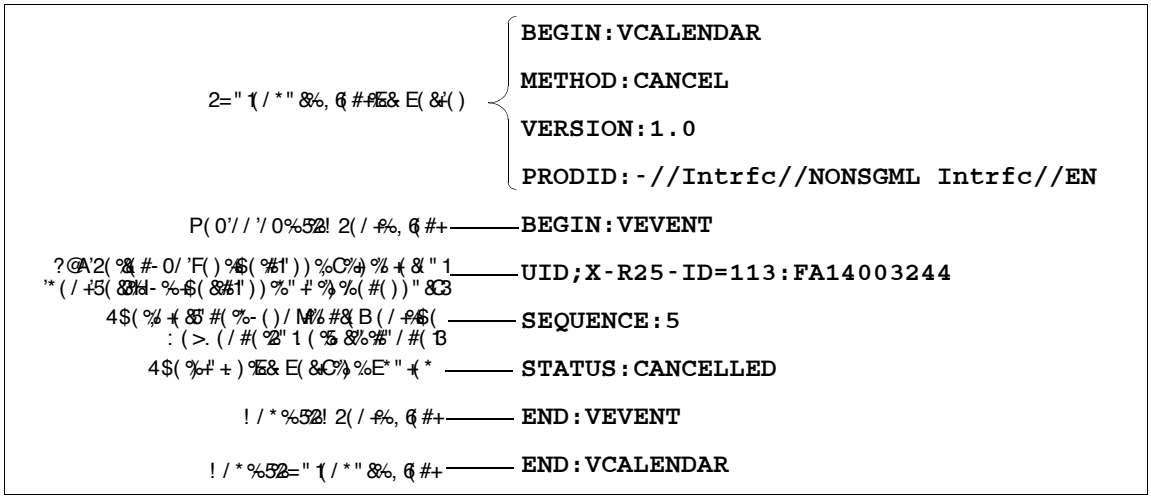
3 vCalendar Files

This chapter shows examples of vCalendar request, cancel, counter, and reply files with property descriptions. The examples don't include all the supported properties. See "vCalendar Properties" beginning on page 22 for a complete description of all properties in the CollegeNET vCalendar implementation.

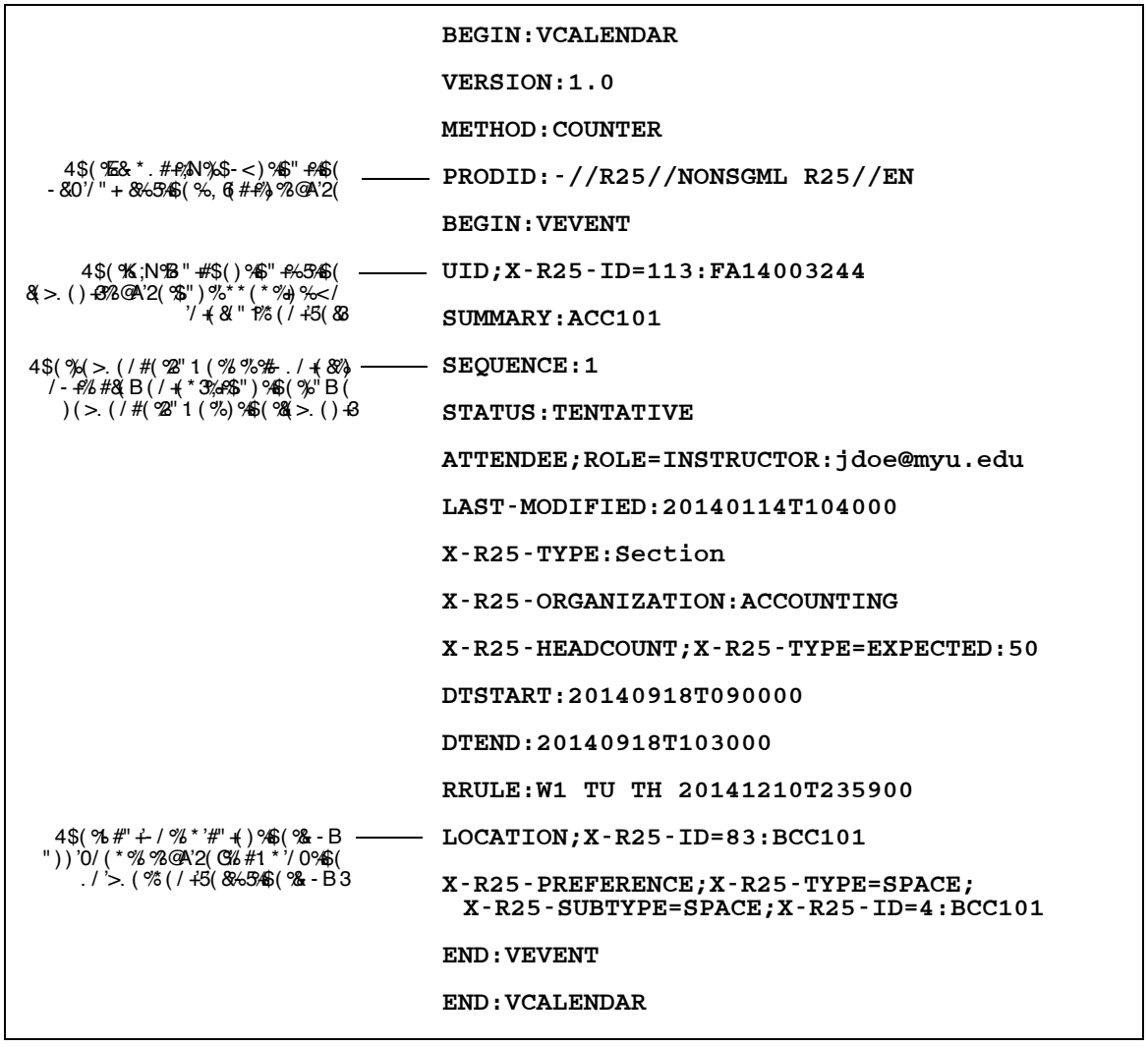
Request file example



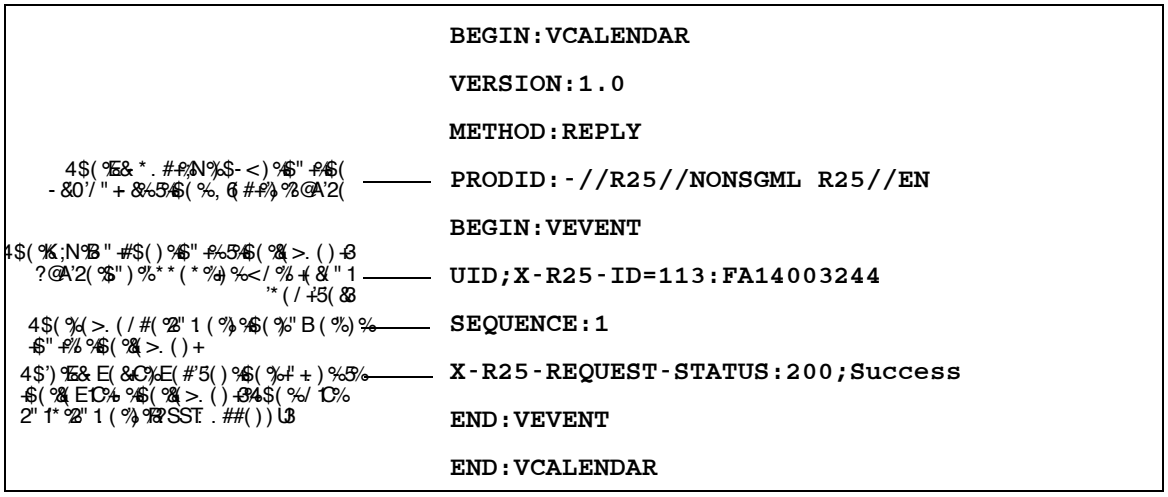
Cancel file example



Counter file example



Reply file example



Request files for more complex classes

vCalendar request files can be used to define classes with more complex scheduling requirements, such as:

- Classes with multiple reservations
- Related classes, such as cross-listed, shared meetings, and so on

Defining a class with multiple reservations

The example on page 20 shows a request file for a class with two reservation meeting patterns: a “class” reservation and a “lab” reservation. To define such a class, the Interface creates three vEvent objects in the request file:

- one for the overall class (the “header” vEvent)
- one for the “class” reservation
- one for the “lab” reservation

The Interface uses these vEvent properties to define data about classes with multiple reservations:

- TRANSP:1 to indicate which vEvent object is the class header and, therefore, that its DTSTART and DTEND values should *not* be considered class reservations by 25Live
- RELATED-TO;X-R25-REL-RESERVATION to link each reservation to the class header
- SUMMARY to specify the class name
- X-R25-RESERVATION-NAME to specify each reservation name
- UID of the class to uniquely identify the class
- UID of the reservation to uniquely identify each reservation; comprised of the UID of the class and an additional identifier (such as 1, 2, and so on)

Example of request file for a class with multiple reservations

```
BEGIN:VCALENDAR
METHOD:REQUEST
VERSION:1.0
PRODID:-//Intrfc//NONSGML Intrfc//EN

BEGIN:VEVENT
UID:FA148765
SUMMARY:CS101-01
X-R25-TITLE:Introduction to Java
CATEGORIES;X-R25-TERM_CODE=1:200710
SEQUENCE:1
TRANSP:1
PRIORITY:0
DCREATED:20140115T104000
LAST-MODIFIED:20140115T104000
DTSTART:20140909T000000
DTEND:20141218T000000
X-R25-TYPE:Section
X-R25-ORGANIZATION:COMPUTER SCIENCE
STATUS:TENTATIVE
END:VEVENT

BEGIN:VEVENT
UID:FA148765/1
SUMMARY:CS101-01
SEQUENCE:1
X-R25-RESERVATION-NAME:Lecture
RELATED-TO;
  X-R25-REL=RESERVATION:FA058765
PRIORITY:0
DCREATED:20140115T104000
LAST-MODIFIED:20140115T104000
DTSTART:20140910T090000
DTEND:20140910T095000
RRULE:W1 TU TH
X-R25-HEADCOUNT;X-R25-TYPE=EXPECTED:60
END:VEVENT

BEGIN:VEVENT
UID:FA148765/2
SUMMARY:CS101-01
SEQUENCE:1
X-R25-RESERVATION-NAME:Lab
RELATED-TO;
  X-R25-REL=RESERVATION:FA058765
PRIORITY:0
DCREATED:20140115T104000
LAST-MODIFIED:20140115T104000
DTSTART:20140911T090000
DTEND:20140911T111500
RRULE:W1 WE
X-R25-HEADCOUNT;X-R25-TYPE=EXPECTED:60
END:VEVENT
END:VCALENDAR
```

Defining a cross-listed class

The example below shows a request file for two cross-listed classes: a math section and a computer science section. To define these classes as cross-listed, the Interface creates a vEvent object for the math section and a vEvent object for the computer science section, and uses the RELATED-TO;X-R25-REL=BINDING-SPACE property to indicate that they must be assigned to the same room. It also puts an X-R25-PRIMARY RESERVATION property in each of the vEvent objects to indicate the "primary class" of the cross-listed group.

Example of request file for cross-listed classes

```
BEGIN:VCALENDAR
METHOD:REQUEST
VERSION:1.0
PRODID:-//Intrfc//NONSGML Intrfc//EN
BEGIN:VEVENT
UID:Fall142425
SUMMARY:MATH360-01
X-R25-TITLE:Mapping Chaos Theory
X-R25-RESERVATION=NAME:Lecture
RELATED-TO;
 X-R25-REL=BINDING-SPACE:Fall051256
X-R25-PRIMARY-RESERVATION:Fall052425
LAST-MODIFIED:20140110T113700
SEQUENCE:1
PRIORITY:0
STATUS:TENTATIVE
X-R25-TYPE:Section
X-R25-ORGANIZATION:MATHEMATICS
DTSTART:20140909T090000
DTEND:20141010909T095000
RRULE:W1 MO WE 20141218T235900
X-R25-HEADCOUNT;X-R25-TYPE=EXPECTED:20
END:VEVENT
BEGIN:VEVENT
UID:Fall141256
SUMMARY:CSCI360-01
X-R25-TITLE:Mapping Chaos Theory
X-R25-RESERVATION=NAME:Lecture
RELATED-TO;
 X-R25-REL=BINDING-SPACE:Fall052425
X-R25-PRIMARY-RESERVATION:Fall052425
LAST-MODIFIED:20140110T113700
SEQUENCE:1
PRIORITY:0
STATUS:TENTATIVE
X-R25-TYPE:Section
X-R25-ORGANIZATION:COMPUTER SCIENCE
DTSTART:20140909T090000
DTEND:20140909T095000
RRULE:W1 MO WE 20141218T235900
X-R25-HEADCOUNT;X-R25-TYPE=EXPECTED:20
END:VEVENT
END:VCALENDAR
```

4 vCalendar Properties

This chapter contains detailed descriptions of all supported vCalendar and vEvent object properties, including:

- The name of the property
- The purpose of the property
- The equivalent 25Live data field, if any
- The property usage syntax with examples
- Which objects (based on method) use the property
- Whether or not the property is required in the objects in which it's used
- In some cases, more information on how the property is used

General information about the supported properties

This is a list of general information about the supported properties:

- All dates and times are local.
- The placement for all vCalendar property values is inline in the data stream.
- A non-standard property parameter, X-R25-ID, defines the internal 25Live value of the corresponding object, for example, **`LOCATION;X-R25-ID=35:Carnegie Hall`**. vCalendar reply and counter files generated by 25Live always contain the X-R25-ID parameter for relevant properties.

vCalendar object properties

The table beginning below describes each of the properties in vCalendar objects. Required properties are shown in bold. Except for the BEGIN and END delimiters, properties are listed in alphabetical order.

If the syntax of a property is shown without examples, the property and property value are exactly as shown.

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
BEGIN	Identifies the beginning of the vCalendar object	none	**Syntax**: BEGIN:VCALENDAR
	Used in:	All objects	
METHOD	Identifies the processing method for all classes in the vCalendar object	none	**Syntax**: METHOD:method type **Example**: METHOD:REQUEST
	Used in:	All objects	
	Possible values for objects generated by the Interface are: · REQUEST - The object defines new or modified classes. · CANCEL- The object defines cancelled classes. Possible values for objects generated by 25Live are: · REPLY - The object is a confirmation response to a request. · COUNTER - The object is a response to a request that contains new or changed room assignments. The method specifies how all classes in the vCalendar object should be processed. So, for example, all classes in a vCalendar object with a CANCEL method are cancelled in the Series25 database when imported. The Interface doesn't read the method in reply and counter files, relying instead on their Interface subdirectory location to determine how they should be processed.		
PRODID	Identifies the product that created the vCalendar object—the Interface or 25Live	none	**Syntax**: PRODID:ISO 9070 value **Example**: PRODID:- / /Intrfc/ /NONSGML Intrfc/ /EN
	Used in:	All objects	
	The value follows the ISO 0970 Formal Public Identifier standard.		

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
VERSION	Identifies the supported vCalendar version	none	**Syntax**: VERSION:1.0
	Used in:	All objects	
END	Identifies the end of the vCalendar object	none	**Syntax:** END:VCALENDAR
	Used in:	All objects	

vEvent object properties

The table beginning below describes each of the properties in vEvent objects. Required properties are shown in bold. Except for the BEGIN and END delimiters, properties are listed in alphabetical order.

If the syntax of a property is shown without examples, the property and property value are exactly as shown. vEvent property parameters are shown in square brackets in the syntax.

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
BEGIN	Identifies the beginning of the vEvent object	none	**Syntax:** BEGIN:VEVENT
	Used in:	All objects	
ATTENDEE	Identifies the class instructor	Event Contact Role, Instructor Instructor must be a 25Live contact	**Syntax**: ATTENDEE[;ROLE=INSTRUCTOR] [;X-R25-ID=R25 identifier]: RFC822 email address **Example**: ATTENDEE;ROLE=INSTRUCTOR: tsmith@myu.edu
	Used in:	request and counter objects	

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
CATEGORIES	Identifies the term being processed	Event Categories: Term Identifier	**Syntax**: CATEGORIES;X-R25-TERM_CODE=1: term code **Example**: CATEGORIES;X-R25-TERM_CODE=1: 201410
	Used in:	request and counter objects	
DCREATED	Identifies the date/time the vEvent object was created by the Interface (normally when the batch export was first run)	Creation Date	**Syntax**: DCREATED:yyyymmddThhmmss **Example**: DCREATED:20140314T153200
	Used in:	request and counter objects	
DESCRIPTION	Identifies descriptive text associated with the class	Event text (Description)	**Syntax**: DESCRIPTION:text **Example**: DESCRIPTION:This class discusses the role of the computer in shaping modern society
	Used in:	request and counter objects	
DTEND	Identifies the end date and end time of the class or the start date and end time of the first class reservation occurrence	Start date and end time of the first occurrence	**Syntax**: DTEND:yyyymmddThhmmss **Example**: DTEND:20140314T090000
	Used in:	request and counter objects	

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
DTSTART	Identifies the start date and start time of the class or the start date and start time of the first class reservation occurrence	Start date and start time of first occurrence	**Syntax**: DTSTART:yyyymmddThhmmss **Example**: DTSTART:20140314T080000
	Used in:	request and counter objects	
	The DTSTART and DTEND properties relate to the actual meeting dates and times of the class or initial class reservation. 25Live also uses the concept of date boundaries to define the first and last possible dates within which an event (class) may occur. These boundary dates are usually inherited from the parent event.		
EVENT-TITLE	Identifies descriptive text associated with the class	Event text (Description)	**Syntax**: EVENT-TITLE:text **Examples**: EVENT-TITLE:This class discusses the role of the computer in shaping modern society
	Used in:	request and counter objects	
LAST-MODIFIED	Identifies the date/time the class or class reservation information was last modified	Last Modified Date	**Syntax**: LAST-MODIFIED:yyyymmddThhmmss **Example**: LAST-MODIFIED:20140204T142700
	Used in:	request and counter objects	
	In a "header" vEvent object, this property is used to indicate the date and time the class information was last modified. When imported into 25Live, this information is included in the Audit Trail history of the class. In a reservation vEvent object, this property is used to indicate the date and time the reservation was last modified. When imported into 25Live, this information updates the Last Modified date for the reservation in the Series25 database which is not visible to the user.		

<table>
<tr><th>Property
(bold = required)</th><th>Purpose</th><th>25Live Data Field</th><th>Syntax and Examples</th></tr>
<tr><td rowspan="3">LOCATION</td><td>Identifies the room assigned to the class</td><td>Assigned Location</td><td>Syntax:
LOCATION[;X-R25-SHARE=1]
[;X-R25-ID=R25 identifier]:room name
Examples:
LOCATION:MM202
LOCATION;X-R25-SHARE=1:ART100</td></tr>
<tr><td>Used in:</td><td colspan="2">request and counter objects</td></tr>
<tr><td colspan="3">The presence of a LOCATION property and value in a request alerts 25Live that the class has a preassigned room.
· 25Live attempts to assign the designated room to the class.
· If 25Live successfully assigns the room and Auto-REPLY has been set for classes of its event state, it automatically generates a reply and sends it to the Interface. If Auto-REPLY hasn't been set for classes of its event state, a 25Live user initiates the generation of the reply.
· If the room is unavailable, 25Live generates a "needs space" To Do. After a 25Live user assigns a room to the class and completes the To Do, he/she initiates the generation of a counter object with the new room assignment which is sent to the Interface which, in turn, updates your SIS.
The absence of a LOCATION property and value in a request alerts 25Live that the class needs a room assignment.
· 25Live generates a "needs space" To Do.
· A 25Live user assigns a room to the class, completes the To Do, and initiates the generation of a counter object with the room assignment which is sent to the Interface which, in turn, updates your SIS.
The X-R25-SHARE=1 parameter indicates that the assigned room can be shared with another class. This allows 25Live to assign the room to two unrelated classes at the same or overlapping times. For example, if a gymnasium can accommodate multiple Physical Education classes at the same time, the "SHARE" parameter allows 25Live to assign the gymnasium to two unrelated PE classes. If the X-R25-SHARE parameter is not used, it indicates that the room can't be shared with other unrelated classes.</td></tr>
<tr><td rowspan="3">PRIORITY</td><td>Identifies the priority of the class</td><td>none</td><td>Syntax:
PRIORITY:0</td></tr>
<tr><td>Used in:</td><td colspan="2">request and counter objects</td></tr>
<tr><td colspan="3">This property is included in all request and counter objects to comply with the vCalendar specification. 25Live doesn't use the value, so it is set to 0 (zero).</td></tr>
</table>

<table>
<tr><th>Property
(bold = required)</th><th>Purpose</th><th>25Live Data Field</th><th>Syntax and Examples</th></tr>
<tr><td rowspan="3">RELATED-TO</td><td>Relates multiple reservations to their associated class
or
Relates cross-listed classes or shared meetings to each other, with a mandate to have the same room assignment</td><td>Event relationship

Binding space assignment relationship</td><td>Syntax:
RELATED-TO[;X-R25-REL=RESERVATION][;X-R25-REL=BINDING-SPACE]:UID
Related events example:
RELATED-TO;X-R25-REL=RESERVATION:SP06004556

Binding space assignment relationship example:
RELATED-TO;X-R25-REL=BINDING-SPACE:SP06654321</td></tr>
<tr><td>Used in:</td><td colspan="2">request and counter objects</td></tr>
<tr><td colspan="3">When used with the X-R25-REL=RESERVATION parameter in a request, this property relates several events (for example, a lecture event and a lab event) to their associated class. The “header” vEvent object describes the class; the vEvent objects describe the related events. When imported into 25Live for the first time, the class is added in the Series25 database with its associated events. See “Example of request file for a class with multiple reservations” on page 20.
When used with the X-R25-REL=BINDING-SPACE parameter in a request, this property establishes a mandatory space assignment relationship among two or more cross-listed classes or shared meetings. This relationship forces 25Live or the Schedule25 Optimizer to place these classes in the same room. Classes linked with a binding space assignment relationship don’t have to have exact dates, days, or times in common. The relationship simply mandates that they be assigned the same room—if one class is reassigned to a different room, all classes in a binding space assignment relationship are automatically reassigned to the same room. If the new room is not available for all members of the group, none of the classes is reassigned to the room. See “Example of request file for cross-listed classes” on page 21.</td></tr>
</table>

<table>
<tr><th>Property
(bold = required)</th><th>Purpose</th><th>25Live Data Field</th><th>Syntax and Examples</th></tr>
<tr><td rowspan="4">RRULE</td><td>Identifies the recurring date/time meeting pattern of the class reservation</td><td>Event dates/times</td><td>Syntax:
RRULE:meeting pattern
Examples:
RRULE:W1 MO WE FR 20141219T180000 //Once a week on Mondays, Wednesdays, and Fridays until 12/19/14 at 6:00 p.m.

RRULE:W1 TH #10 //Once a week on Thursdays for 10 occurrences</td></tr>
<tr><td>Used in:</td><td colspan="2">request and counter objects</td></tr>
<tr><td colspan="3">This property is used to define a meeting pattern for a series of class reservation meetings. It is used with the DTSTART and DTEND properties which define the start and end date/times of the initial reservation occurrence.
Currently, vCalendar supports the inclusion of only one RRULE per vEvent object, so if a class has multiple recurring reservation patterns, one vEvent object is included for each, and the RELATED-TO reservation property (see page 28) is used to link them.</td></tr>
<tr><td></td></tr>
<tr><td rowspan="4">SEQUENCE</td><td>Identifies the number of times the shared class data has been changed</td><td>Version Number</td><td>Syntax:
SEQUENCE:sequence number
Example:
SEQUENCE:17</td></tr>
<tr><td>Used in:</td><td colspan="2">All objects
Only incremented by the Interface, never by 25Live.</td></tr>
<tr><td colspan="3">This property identifies the revision number of the vEvent object. On import, 25Live ignores vEvent objects in request files with a sequence value that is less than or equal to the value in the current 25Live event record of the imported class. If there are several related vEvent objects (for example, a header and several reservations) the same sequence value is included in each. The Interface initializes the sequence value of a class to 1 (one) when it exports the class for the first time. Each time the shared class data is modified in the SIS, the Interface reads the sequence value in the Series25 database and increments its value by 1 (one) in the request object it generates for the class. The Interface doesn’t increment the sequence number in cancel objects.</td></tr>
<tr><td></td></tr>
</table>

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
STATUS	Defines the current status of the class—TENTATIVE for active classes (unless changed in the Interface Configuration file) or CANCELLED for cancelled classes	Event State	**Syntax**: STATUS:event state **Example**: STATUS:TENTATIVE
	Used in:	request, cancel, and counter objects	
SUMMARY	Specifies the name of the class	Event Name	**Syntax**: SUMMARY:event name **Example**: SUMMARY:CS102A
	Used in:	request and counter objects	
	On import into 25Live, the SUMMARY value is entered into the Event Name field in 25Live for the class (only the first 40 characters are entered).		
TRANSP	Identifies the "header" vEvent object	none	**Syntax**: TRANSP:1
	Used in:	request and counter objects	
	This property is used in conjunction with the RELATED-TO;X-R25-REL=RESERVATION property/parameter described on page 28 to identify the "header" vEvent object. It indicates that the vEvent object is "transparent," that is, that its start and end dates should not be treated as actual reservation dates and a reservation should not be created for them in 25Live. The expectation is that the start and end dates in the subsequent vEvent objects linked to the "header" using the RELATED-TO property contain the actual reservation meeting pattern(s). See "Example of request file for a class with multiple reservations" on page 20. The absence of the property is equivalent to TRANSP:0 and means that the dates defined in the vEvent object represent the actual reservation dates of the class.		

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
UID	Identifies the unique identifier of the class or class reservation	none (but is used internally)	**Syntax**: UID[;X-R25-ID=R25 identifier]:UID **Example**: UID:F05CS102A
	Used in:	request, cancel, and counter objects	
	The UID property value is made up of the term identifier, CRN, and (if applicable) meeting pattern number. In cases where several vEvent objects are required to represent a single class, each vEvent object has a unique UID. For example, F05CS102A for the "header," F05CS102A/1 for the lecture reservation, and F05CS102A/2 for the lab reservation.		
X-R25-HEADCOUNT	Specifies the expected or registered head count of the class or class reservation	Expected or Registered Head Count	**Syntax**: X-R25-HEADCOUNT [;X-R25-TYPE=headcount type]:number **Example**: X-R25-HEADCOUNT;X-R25-TYPE=EXPECTED:50
	Used in:	request and counter objects	
X-R25-ORGANIZATION	Identifies the primary department sponsoring the class	Primary Organization	**Syntax**: X-R25-ORGANIZATION [;X-R25-PRIMARY=1] [;X-R25-ID=R25 identifier]: department name **Example**: X-R25-ORGANIZATION;X-R25-PRIMARY=1:ENGLISH
	Used in:	request and counter objects; "header" vEvent object only	

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
X-R25-PREFERENCE	Identifies the room, campus partition, or room feature preferences of the class	Space Preferences	**Syntax**: X-R25-PREFERENCE[;X-R25-TYPE=SPACE][;X-R25-SUBTYPE=space preference type][;X-R25-ID=R25 identifier]:value **Room preference example**: X-R25-PREFERENCE;X-R25-TYPE=SPACE;X-R25-SUBTYPE=SPACE:BCC101 **Partition preference example:** X-R25-PREFERENCE;X-R25-TYPE=SPACE;X-R25-SUBTYPE=PARTITION:Bothwell Building **Room feature preference example:** X-R25-PREFERENCE;X-R25-TYPE=SPACE;X-R25-SUBTYPE=FEATURE:Tiered Seating
	Used in:	request and counter objects	
X-R25-PRIMARY-RESERVATION	Identifies the primary class of a cross-listed or shared meeting group	Primary Reservation	**Syntax**: X-R25-PRIMARY-RESERVATION:UID of primary class **Example**: X-R25-PRIMARY-RESERVATION:F05004556
	Used in:	request and counter objects	
	This property is included in the vEvent object of each member of the cross-listed or shared meeting group.		
X-R25-REQUEST-STATUS	Specifies the status of a vCalendar data exchange transaction	none	**Syntax**: X-R25-REQUEST-STATUS:200;Success
	Used in:	reply objects	
	This property indicates the status code returned from 25Live in response to a request or cancel. Only the 200 “success” status is supported. If a vCalendar or vEvent object generated by the Interface contains errors, 25Live moves the object to a file in the specified vCalendar import errors subdirectory. *The request status is not used to return error information.*		

Property (bold = required)	**Purpose**	**25Live Data Field**	**Syntax and Examples**
X-R25-RESERVATION-NAME	Specifies the name of a class reservation	Reservation Name	**Syntax**: X-R25-RESERVATION-NAME:name **Example**: X-R25-RESERVATION-NAME:Class Meeting
	Used in:	request and counter objects	
	On import into 25Live, the X-R25-RESERVATION value is entered into the Reservation Name field in 25Live for the class (only the first 40 characters are entered).		
X-R25-TITLE	Specifies the title of the class	Event Title	**Syntax**: X-R25-TITLE:event title **Example**: X-R25-TITLE:CS102A - Introduction to Java
	Used in:	request and counter objects; “header” vEvent object only	
	On import into 25Live, the X-R25-TITLE value is entered into the Event Title field in 25Live for the class (only the first 240 characters are entered).		
X-R25-TYPE	Identifies the event type of the class	Event Type	**Syntax**: X-R25-TYPE [;X-R25-ID=R25 identifier]:event type **Example**: X-R25-TYPE:Section
	Used in:	request and counter objects	
	The property is included in all new request files generated by the Interface. The value must conform to your Event Type Hierarchy to help ensure that the class can be properly routed into your Series25 database on import.		
END	Identifies the end of the vEvent object	none	**Syntax**: END:VEVENT
	Used in:	All objects	