

## Calendaring and Scheduling 101: How to Use IBM Lotus Notes and Domino Calendar and Scheduling in your Applications

**Session ID:** AD301  
**Session Track(s):** Track Two: Application Development

**Speaker Name:** Youcef Bennour  
**Speaker Company:** IBM

**Session Abstract:** Come learn about the inner workings of the Lotus Notes and Domino Calendar and Scheduling (C&S) system. This session will provide information to help you use Calendar and Scheduling in your applications, including how calendar events are represented in Lotus Notes, what happens when creating or updating a calendar event, and the overall workflow process. We'll also include a sneak preview of the future C&S APIs now under development.

### Transcription:

BENNOUR: Welcome, everybody. Welcome to the Calendaring and Scheduling 101 session. My name is Youcef Bennour. I'm with IBM, and I'm located in Westford, Massachusetts. Before I forget, I was asked to ask you guys to fill out the forms and return them to the people outside at the end of the session.

This is intended to be an intermediate to advanced session.

It is meant for developers that want to create and edit CNS data as part of their Notes application.

As part of the agenda, I'll go through a demo that shows how

to, how basically we create a meeting, update the chair's calendar and send invitations to participants. And then next I'll go through the Lotuscript Schema.

And then we'll walk through the Lotuscript, implement the demo. And we'll talk a little bit at the end, towards the end, about possible futures about CAPI that we're currently planning. And then we'll open the floor for Q&A.

Let's go for the demo now. As part of the demo, this is a teamroom database to which we added the item which is a set up meeting. The leader can click on it and invokes this dialog box.

And we can go and change the date. Make it the 25th. We can change the time to be a little bit later, two p.m., and I'll change the end time to be...well, that's too much...to be five p.m.

I'll set the location to DL north and hemisphere. We'll set the subject to, this is my demo for Lotusphere. And we'll set, this is my description for the description.

Now, the required to field has already been populated with the members of the teamroom. I'm just going to move a few of the names to the other fields. I'm going to take the last two and move them into the FYI. These are the people

that need to know about the meeting that are not required to be there. And I'll take the next two and put them as optional invitees. And then click okay.

Now, if I go into my mail and then go into my calendar, here's the meeting that we just created and without going through the Notes application, the Notes CNS application. And we can verify that all the things that we entered are all there.

I'm going to go and open Notes, the Notes client for one of the participants. And let's open it, this guy. Boy, is it slow. Come on.

And here's the invitation notice that we just sent. If we open that. And we accept the meeting, then this gets added to our, to this person's calendar. And there is the meeting in his calendar now.

If I go back to the chair's calendar, the chair's mail, I will see, I will accept notice being delivered. This, that ends the demo at this point.

Most of the information that I'm going to go over can be found in a document at the following URL. The purpose of the document was, that is, to assist in the development of products that interact with the CNS feature.

All CNS objects are stored in the [bail] file. And we have calendar events, task events, reservation events and CNS work flow messages. And the form is used to distinguish which type of CNS object the entry is.

There are five types of calendar events and two types of task events. And an item, an appointment type is used to distinguish which type of calendar or task event the entry is.

This item was added in R5. It addresses a problem where an event can happen more than one UNID. And that can happen if User A prefers RTF format and User B prefers MIME format. And when we try to deliver notices to them, we create two different notes in mailbox which basically creates two universal note IDs for the event. So to get around that, we created the app UNID item, and we are storing the UNID of, your general UNID, of the document in it.

All calendar events that needs to get displayed in the calendar view needs to have calendar daytime set to date and time where it should appear in the calendar. Object, CNS object that are not, that don't have that item will only be seen in the all calendar entries in Version 7 and up or meetings, the meetings in Version 5 and 6 or in the all document view.

Single instance events are scheduled to happen only once, and they're described as only one note document. Repeating event on the other hand happens more than once, and they're described, used at least two notes in a parent-child relationship, or the repeat is the item that indicates if an event is repeating or not.

The repeat dates and repeat instance dates are unique to repeating events. Repeat instance dates have a snapshot of the original date and times for the event, and repeat dates are the current date and time for the event.

Repeat UI fields are used to specify a repeat rule. The repeat rule is used to first compute the very first, the original date and time for the event.

Now, the child document has, describes a run of consecutive days in which all the items in the events are the same. It has both repeat instance dates and repeat dates, just the same as the parent, but they only contain start dates and time only for the run of instances that the child document describes.

The child document is tied to the parent using the response, or reference list item. And child documents have the calendar daytime set to list of dates where the items should

appear in the calendar.

As an example, I just want to show basically what happened when you create repeating event internally in Notes. The creation of the two notes, one is the parent. The other one is the child. And what happened was, you edit one instance of that, in that run.

We're going to create an event that is repeating five days in a row. And the start date for the repeating set is set to the 22nd, and the starting time for the event is 3 p.m.

The initial, initially we create only a parent note, and we set all of the UI repeat fields. Well, that one, when you try to save the note, then we generate the repeat instance dates and repeat dates. And the child note gets created.

And the child note has the same type of information. It has repeat instance dates, repeat dates, calendar daytime. There is the list of the daytime where it should appear in the calendar view, and it has start time and end times.

Now we are going to change the third instance, the one on the 24th to start at 4 p.m., and this is what happened. If we change the start date time on it, what happened is, first the parent, the repeat dates in the parent gets updated. And the repeat instance dates is kept unchanged. And then

the child note is splitting to three notes.

The first document basically describes the first two instances in which all the items are the same. The second one describes the exception that we just made. And the third one describes the last two instances. You can think of this as run length, including of the instances.

Event IDs, single instance event IDs, like I said before, are identified by their app UNID field. And a particular instance in repeating event is identified by key value pair, the parent UNID and its initial start date time.

Work flow processes. They're really relevant only to meeting calendar events, where more than one party are involved.

They're implemented by notices sent back and forth between the chair and the participants. And the notices basically have, the notice document has a field, an item that is notice type. That describes what type of notice is being sent.

These are the type, the notice type sent by the chair. The chair can say, invite people, can cancel the meeting, can reschedule the meeting and update information.

The one...notice type sent by participants are, the participant can only accept an invitation or decline it or delegate to somebody else, or counter-propose a request. Propose a new time.

Try to describe this, I rather go through animation to show basically what happens when you try to create an event and how the notices get sent. First we create the note, the document. And then we set the start time, start date, time and daytime.

Next we're going to invite one person to the meeting. Now, I'm...we're setting the subject, description and location. This is just collecting basic information about the meeting.

Now we're about to send the notice to the invitees. One thing that we do is, we save the subject as entered by the chair in the topic field, and then we alter the subject to be prefixed by invitation. Once that goes there, now we're changing the document to be a notice document. And the notice type is of type I for invite. Once that happened, an invitation notice is sent to the participant.

Now, as part of saving the document, after the notice has been mailed, we morph back the notice document into a calendar document. And that is done by changing the form to appointment, and we are setting the calendar daytime to

where it should appear in our calendar. And we are updating our visit time info, and we're restoring the subject to what it used to be.

And then the topic and notice type are removed from the document, and the document is saved to the calendar. There are a couple of more steps if you're dealing with a repeating event.

The last two steps basically, what happened is, you generate the instance dates. And then when you save the document, the child document is created. Then everything is saved to your mail file. The child document would have the calendar daytime, and it will appear in your calendar.

Now, now the chairman wants to invite another person, and that person cannot make it on the specified time. So it's...he's going to have to reschedule the meeting with the first participant and send an invite notice for this new one.

So the very first thing he does is change the time to where basically, when the new person is free. And then he invites the person. Now, we have to remove the data that is pertinent only to us.

The current daytime item has to be removed, otherwise when

the notice gets to the participant, they'll see it in their calendar without having to do anything. So that is going to get removed, and our busy time info is also removed because it's ours here. Then we morph the document into a notice document, and this one is a re-scheduled notice document.

And then rescheduled notice is sent to the original participant first. Next we'll have to send an invitation to the new participant. So we set the notice type to I for invitation. We change the subject item appropriately, and a notice goes to Susan, who is the new participant.

After this now, we're going to morph the document, the notice document back to a calendar document. And we're going to set the calendar daytime to new time so that it would appear in the right spot in our calendar.

We'll update also our busy time info, and we'll restore the subject to what it used to be. And as before, notice type and topic are removed from the document, and the document is saved to the calendar.

Now, a participant sees a notice getting deposit in their mailbox. They open it. And you guys have seen this already before. This is what you see when you open an invitation.

Now, since this participant is [Barry Amin], [Barry Amin]

can either accept, decline, counter-propose or delegate the meeting. We'll go through the process of accepting first.

The subject is generated using the data in the topic item, and we prefix that with accepted. And we turn, and we set the notice type to be accept notice. And the accept notice is sent back to the chair, telling them that we have accepted the meeting.

And then we morph back the, we morph the notice into a calendar object by setting the form to appointment. We set the calendar daytime to the start daytime of the meeting that will appear in our calendar now. And we update our own busy time info. And we restore the subject to what the chair has originally inputted. And then the note is saved into the calendar.

Same thing when we delegate. It's close to...the process is almost always the same. You do some changes to the note. You send notices back, and you update your calendar if you need to.

To delegate, there are four fields that get, either created in the note there or updated if they're already on the note.

Delegate to list is the list of the delegates in the order they were delegated to.

The delegator is the owner of the calendar, the current one.

And the delegatee is the person we are going to delegate to. And the original delegator is the very first guy that started the delegation. The subject gets changed to invitation delegated as a topic. The notice gets set to L, and a delegate to delegatee notice is sent.

Next we're going to inform the chair that the, that we have delegated the, to somebody else. The subject is changed appropriately again.

The notice type is changed to D for delegate for the chair, and the notice is sent to the chair informing him of the delegation. The chair uses the notice document to type the delegator to the delegatee. That way any new updates are sent to the delegatee.

The process when we decline is really simple. You just set the notice type to say that this is a decline notice. You build, you set the subject to say decline. And then the, whatever the subject, the way it was entered before. And the notice gets sent to the chair. And then subject gets set to whatever was entered by the chair, and the document is just saved in the in box.

When we counter-propose, we're proposing new time. And these are the fields that you have to set in order to

propose the new time.

the subject gets changed again to say counter with the, whatever the chair has entered before, and the notice type for calendar is T. And then a notice gets sent to the chair. And then the document just, is just saved in your in box.

The chair can use this item to prevent people from answering, basically sending responses to him. This is useful when you set up a big meeting, where you don't want people basically to answer you, otherwise your mailbox would be flooded with notices.

The chair can also prevent counter, or delegation. These are the two items, if they're set to one, then you're not allowing delegation or the calendaring. Any other values, including not being there basically, just says I won't cover it or delegation.

A participant, when they decline or they delegate, can still ask the chair to keep them posted of updates about the meeting. Setting this field to one signals that you still want further updates.

These are the fields that are used to determine which instances to apply an operation to. The first one is the

one that everybody must be familiar with. Basically those are the thing that, when you're editing an instance, you get this dialog box that asks you at the end when you try to save, are you doing this operation to all instances, only in this instance, or this one and previous, or this one and futures. The next three ones are, I think, are new with R6 and up.

You can specify exactly which instances you want the operation to be, to apply the operation to. Reschedule instance dates is the list of dates of, the original list of dates of instances that you want to change. The schedule end daytimes and start times are the times that you want the new instances to have if it's a reschedule for example.

Sequence number track changes in scheduling meeting time. It's only update, it is only updated when we do a reschedule. Update sheets tracks own daytime changes. Item changes. Things like subject, description or anything else. And watch item sequence list is, tracks basically which item have been changed.

Now, we're going to go through, walk through the Lotuscript codes that I built to have the demo that you saw at the beginning. The set up meeting, the hot spot basically invokes this script.

This script just creates an instance of CS add meeting, which is a Lotuscript class that implements all of the details of what you saw.

Anything the object, just edits the object and creates a note. Opens a mail file. Creates a note, and then populates the items by using the four values.

And then this call here is the call that, that invokes the dialog box in which we can change stuff. And if we say okay on the dialog box, then we call this method to save and send the invitation.

When we create the class, we just initialize three instance variables. When we call in it, we open the mail file. We make a note. We create a note in it, and we initialize the items.

This method basically initializes some of the basic items like the form, the appointment type, the version, the sequence num, update seek and the like. And also we set the chair and the [on] chair principal to the user name. The person who started the session.

This method only initialized the date/time items. Most of the code in the first part is really trying to select the next 15-minutes slot in the, after the time.

And then most of the code is just saying, use this as the start, the original start time. And start date, start time.

And then add the default duration for the meeting that will be an hour, and compute basically the end date times.

That's the call that gave us the dialog box.

And this is the code that basically saves, and sends an invitation. We do go and set the default alarm settings. You can set the alarm to be notified when it's almost time for the meeting to happen. We prepare the invite notice. Then we send the invitation. And we revert to the meeting document, and then we save it.

Alarm, to enable an alarm, you set the dollar sign alarm item to one. And then you can, you can specify a description to be displayed when the alarm triggers. And you can set the time when the alarm should trigger. And here we, I just said half an hour before the meeting.

This is preparing the invite notice. We remove, like I said, we remove if we have a calendar daytime entry item in the document that would remove it. We remove our visit time info. We set the notice type to I and the form to notice.

We set the icon to be the invitation icon. And we change, we basically change the subject to have the right format.

The thing that says invited and there, whatever the chair has input and then a formatted daytime.

This method just formats the daytime to the appropriate format. And after the notice is sent, we revert to the meeting, to...we morph the document back from a notice back to calendar event.

You can see here what we saw in the animation. Calendar daytime gets set to the start time. The...we remove the notice type item. We update the visit time info. We set the icon to meeting icon. We set the subject from the topic.

We just restore the subject to what used to be. And we remove the topic item. And we also just empty the send to fields and the mail, the mail fields basically. The send to, copy to, and FYI.

We are currently planning a CAPI for the CNS libraries that exist, or that the [tempered] code uses. The goal is to define a set of API for high-level functionality. Like I want to create an event, and set the time for this, set the description and so on. And then when you see, save, everything should happen without you having to worry about anything else. The notices should be sent and everything.

This is the thing that we're thinking about right now. We have CNS implementation libraries that exist already. We're going to try to abstract into a CAPI, and then we are planning on exposing that CAPI through back-end classes to note applications. And we have also plans for having web services basically using that CAPI.

We are planning two interfaces right now. We're thinking about calendar service interface that works at the calendar level. You ask that calendar interface basically to create a meeting for you.

And you will get an object which is a lower interface, and you can interact with that interface. You can set the, that interface will have a bunch of set properties. And then when you save that...when you call a save basically, what happened is, it should hide all of the details of how the notices get built, sent, and the fourth.

We need at least one API function to at least start the process and get one calendar object, service interface basically allocated and initialized.

Well, now is the time to let us know your needs and requests. We want your input, since we are still in the planning stages for this CAPI. Otherwise we'll just do, and try to abstract the best of our ability.

And we are basically looking at you, basically for getting, giving us input of what you need and what you rather see there. Now, questions and answers. Yes.

QUESTION: [INAUDIBLE].

M: Right now, the work is planned for R8. I'm not sure if they're going to allow back work of it. Yes.

QUESTION: [INAUDIBLE].

M: I'm sorry, I couldn't hear you.

QUESTION: [INAUDIBLE].

QUESTION: [INAUDIBLE].

M: Yes.

QUESTION: Can the people asking the question come to the microphone? We have microphones up in the front here.

M: I'm sorry, I'm just coming out of a cold, and my eyes, my ears are still clogged.

QUESTION: Yes. With the new ghosted entries, and you said an invite, it automatically appears in your calendar view. How do we do that? Is there a calendar...

M: That's because basically it's just looking for items that are calendar and scheduling objects.

QUESTION: Okay.

M: It's just listing basically documents.

QUESTION: Okay. So basically once you turn that option on, any calendar...

M: Yes...

QUESTION: ...every option...

M: Yes. That is on by default. That's how we can look at them if they're not in your calendar view.

QUESTION: Okay, thanks.

QUESTION: Can you flash that URL again? I didn't get to write it all down.

M: Sure. That would be faster. Is that good enough?

QUESTION: Is example code that you had there for the team work database going to be available for download anywhere?

M: I'll make it available, because the thing that

I submitted is complete different.

QUESTION: Okay.

M: The thing I submitted was using the wrong things. And they weren't published basically in...

QUESTION: Lovely.

M: You guys cannot use them, and they're not supported. Even if you do, basically you're on your own.

QUESTION: Okay. But they'll be down there with the slides. Correct?

M: Yes. I'll make everything available.

QUESTION: Okay. Thank you.

QUESTION: When.

M: As soon as I am back at Massachusetts, I guess.

QUESTION: [INAUDIBLE].

QUESTION: I have a question.

M: Yes.

QUESTION: Is there any thought of integrating calendar and scheduling with outside products, CRMs, Outlook, Web?

M: Right now we don't have any plans for that. But make the suggestion, and we take it into consideration.

QUESTION: Okay. Okay. Can I speak to you afterwards.

M: Sure.

QUESTION: [INAUDIBLE].

M: H'mm?

QUESTION: [INAUDIBLE].

M: Oh, okay.

QUESTION: [INAUDIBLE].

M: Thank you.

QUESTION: Will you be [INAUDIBLE] out at any labs so I can talk...[INAUDIBLE].

AD301

M: Yes. I'll be at the developers' lab.

QUESTION: When?

M: After this. After lunch. And I'll be there until 6:00 or 5:30, whenever they close. All right?

[END OF SEGMENT]