****

**SIF Change Proposal: Bundling and transporting multiple SIF Events in a single “*Events*” Message**

**Submitted by: Infrastructure Functional Group**

**Version: v5.1**

**Date Submitted: July 23rd, 2008**

**Last Revised: Feb 1, 2012**

Table of Contents

[Business Case 3](#_Toc315727008)

[Timeline 3](#_Toc315727009)

[Introduction 3](#_Toc315727010)

[FAQ 3](#_Toc315727011)

[Goals of the Solution 7](#_Toc315727012)

[Details of the Solution 7](#_Toc315727013)

[Detailed Header Analysis 9](#_Toc315727014)

[ZIS *Events* Message Processing 10](#_Toc315727015)

[Minimizing Latency 11](#_Toc315727016)

[Bundle Processing 11](#_Toc315727017)

[XML Examples 11](#_Toc315727018)

|  |  |  |  |
| --- | --- | --- | --- |
| Draft Specification Document Version Control | | | |
| Version | Date: | Author/Organization: | Comments |
| V1.0 | July 23, 2008 | Andrew Elmhorst | Initial Document |
| V2.0 | May 3, 2011 | Xavier Wiechers & Zachery Jensen | Modified document to update the Business Case and Timeline as well as to introduce the proposed solution |
| V3.0 | May 13, 2011 | Xavier Wiechers | Made a few modifications from the feedback from the Infrastructure meeting of May 4 |
| V4.0 | Aug 16, 2011 | Xavier Wiechers | Generalize the document to become Bundled Messages as opposed to specific to Events. |
| V4.2 | Aug 18, 2011 | Ron Kleinman | Fleshed out the design |
| V4.3 | Aug 19, 2011 | John Lovell | Provided example to be converted to the SOAP transport. Other tweaks as I felt necessary. |
| V4.4 | Nov 1, 2011 | Ron Kleinman | Added changes for Specification |
| V4.5 | Dec 7, 2011 | Ron Kleinman | Added comparison with alternative “multi-event” Event message |
| V5.0 | Jan 30, 2012 | Ron Kleinman | Responded to Web Service interface problems with packaging a Message Bundle. The facility now bundles only Events |
| V5.1 | Feb 2, 2012 | Ron Kleinman | Reflects final Infrastructure Team Approval of Events Message |

**Change Proposal**

Business Case

In large implementations of SIF we are experiencing scalability issues with processing excessive numbers of small SIF Event messages. These issues are most noticeable during burst operations such as taking attendance, during initial enrollment, year-end operations etc. Sending many changes from one agent to another via separate SIF Event messages can be inefficient. The Change Event messages in particular tend to be small and the SIF Ack overhead becomes a large part of the message payload (sometimes as large as 40%). Additionally, waiting for an acknowledgment before sending the next message in the queue means any: delay in persisting messages or latency in transferring messages, compounds to greatly lengthen the time required to transfer any large collection of messages.

Both the AU and UK in particular have been active requesters for increased scalability.

Timeline

This change will be designed and implemented as part of the SIF v3.0 release. However despite being in a major release, this proposal endeavors to introduce “Events” message functionality in such a way that it will maintain backwards compatibility and seamlessly integrate into Zones which have existing agents that do not send or receive such messages.

Introduction

Bundling multiple Event messages in a form where they can be transmitted and Ack’d as if they were a single message, will allow both the client and the ZIS to maximize the use of the recipient’s maximum buffer size as well as minimize the number of required SIF Acks. While this bundling capability was originally conceived of as pertaining to Event messages only (where it has its biggest impact on improving scalability) it was later generalized to include all types of messages. *However based upon feedback during the design process, “bundling” has again been restricted to apply only to Event messages.*

FAQ

Many questions regarding bundling messages of any type have been asked. Here are some of the most common, complete with brief answers

* What exactly is a bundle of Event messages and how is it packaged?

**A series of Event Messages in a transmission queue may be concatenated into an “*Events*” message, and sent out as a single message. The *Events* message has all the normal header elements of any SIF message. Its contents are the complete (header and data) Event messages which were packed into it.**

* Are there any restrictions on the sorts of Events that may be bundled into an *Events* message?

**No. They can be of different Event types (Add, Change or Delete) and be describing changes to objects of different object types (StudentPersonal, Assessment or GradeBook).**

* Why are only Events bundled and not all types of messages?

**Originally the idea was to create a general SIF message bundling facility, but the design ran into a problem regarding the new Web Service infrastructure stack. Essentially any Push Mode Agent supporting Web services would have to support the new “Here is a Bundle” method as the only method invocation in a newly defined “Bundle Service”.**

**So far, so good.**

**But a problem arose because any code supporting this method would have to pop open the received Bundle, and it would then be faced with processing a series of internal SIF messages which reduplicated what would be expected from a random set of method invocations that potentially spanned all the other services which that Agent supported. This would clearly break SIF service interface independence and was deemed by one Agent vendor as hard to implement.**

**By bundling only Event messages, the Web Service interface that supports receiving them could be extended to support a new “*Events*” message and no other SIF service would be affected. Since the bundling scheme was originally introduced to deal with a flood of Event messages during times like the end of a reporting period, restricting bundling to only Event messages was deemed to not be a significant detriment.**

* Is “*Events*” a new kind of message?

**Yes. While it is basically a container for multiple queued up Event messages, it should be considered another message type since it is Ack’d in a manner similar to any other SIF message. Individual Event messages within an *Events* bundle are not individually Ack’d.**

* Can a sender issue an *Events* Message or only the ZIS?

**Both Agents and the ZIS can issue an *Events* Message.**

* Does the ZIS have to “unbundle” all arriving *Events* messages?

**Yes for many reasons. For example internal Event messages could be reporting changes to objects of different types, and so need to be sent to different destinations or have different security requirements.**

* Does the ZIS have to “bundle” Event messages for a subscriber who can deal with *Events*?

**The requirement for a ZIS to support the new *Events* message in SIF v2.6 will be mandatory and includes the ability to both unbundle and bundle Event messages. However for a variety of reasons (ex: low network traffic), the requirement to bundle Event messages in any specific instance will be optional.**

* How many Event messages can be bundled in an *Events* message before sending?

**All Event messages in the recipient queue can be bundled, up to the limit of whether the *Events* message will fit into the recipient’s maximum buffer size. The minimum number of Event messages in an *Events* Message is one.**

* How long does the ZIS wait before shipping the current set of queued up Event messages?

**This is likely an Administrator-settable ZIS parameter. See the “Latency” section below.**

* Does / should the ZIS wait till the Event bundle has reached its maximum size?

**Not generally (for example latency factors may intervene), but it can**.

* How is Message ordering maintained?

**This was not an issue for the Message Bundle design, where the strategy was basically “first come first bundled”. Now that only Event message are bundled, the question arises of whether queued up Event Messages can be combined into a single *Events* message, even if they are separated in the queue by non-Event messages. For infrastructure release SIF v2.6 the answer is “no”.**

**This restriction may be revisited in subsequent releases. It is anticipated the change, if it occurs, will be invisible to the subscribing Agents.**

* Can individual Event messages for the same object coming from 2 different publishers be bundled?

**Yes. An arriving *Events* message is always broken up into individual Event messages by the ZIS and processed. These Event messages in turn may be added to *Events* messages being built in the message queue of one or more Agents or queued up as individual Event messages ... or both.**

* Can an *Events* message ever be bundled into a larger *Events* Message?

**No! While theoretically possible, bundling is not currently defined as a recursive operation. It would be very difficult to define all the “corner cases”, it would add needless complexity for those applications receiving an *Events* message, and there seems no compelling reason to allow it. An *Events* message contains only Event Messages.**

* Can an *Events* message be packetized?

**No. An *Events* message is conceptually a “multi-Event packet”. Allowing it to be packetized would add complexity but not aid Zone scalability.**

* Can an *Events* message sent from an Agent to the ZIS have a Destination ID?

**No. The Event messages it contains are independent and (if created by an Agent) may be destined for different destinations. However the internal Event messages can have Destination IDs (which may be different from each other).**

* How exactly are a series of Event Messages “packed” in an *Events* Message?

**The packing is done by simple concatenation within the Events / SIF\_Events element. There are no additional packing-specific elements (like “EventBreak”) located in the body of the data, or in the *Events* header (like “NumberOfEvents”). Unpacking is done whenever the end of an internal Event / SIF\_Event element is reached.**

* What happens if the receiver detects an error with the *Events* Message itself (like it doesn’t support that message type)?

**An *Events* Message should never be routed to an Agent which does not support it, since this situation represents some sort of provisioning error. This means a transport level problem with an *Events* message should be treated the same as with any other message type. The *Events* is rejected with a synchronous Error and possibly a LogEntry is posted.**

**On receiving the error, the sender of the *Events* then frees up all Event messages in the recipient’s queue that were part of the *Events* message ... it does not respond to the error by unpacking the *Events* and re-sending only the first internal Event. For further details, see the *Error Handling* section below.**

* What happens if, as it is unpacking and processing an Event within the *Events* message, the receiver detects an error?

**The erroneous internal Event in question should be processed exactly as it would have been if it had arrived individually (except no individual HTTP ACK will be sent).**

* Will *Events* be supported over both the HTTP/S and SOAP transports?

**Yes. A special SIF v2.6 Infrastructure release has been created to package this extension to the existing message set.**

* How does a sender determine whether the recipient supports the *Events* message?

**There are only two cases to be dealt with although all three are covered below**.

* + **How does an Agent determine whether the ZIS supports *Events*?**

The Infrastructure Version of the ZIS should be known in advance and can be retrieved after Registration if needed. *Events* message support will be mandatory for SIF-certified ZIS products as of the SIF v2.6 release.

* + **How does the ZIS determine whether the Agent supports *Events*?**

The Agent indicates whether it does or does not support reception of the Events Message at Registration time. From SIF v2.6 on, any Agent may support receiving and / or sending *Events* messages. From SIF v3.0 on, all Agents will be required to support receiving *Events* Messages.

* + **How does an Agent determine whether its subscribers support *Events* messages?**

The sender is independent of whether the intended destination supports *Events* messages, because the ZIS can break open the *Events* if necessary to successfully deliver the individual Event messages it contains to the destination Agent. In fact, this decoupling of sender and receiver is critical since any Event message must be capable of going to multiple recipients (only some of whom might support the reception of *Events* messages).

This proposal will expand on these answers in the following sections.

Goals of the Solution

*Events* Messages satisfy the following design goals:

1. Provide complete backwards compatibility to allow agents that do not support *Events* to continue to send and receive SIF Event messages exactly as they did before
2. Keep the interface simple so that agents and ZISs can easily implement Event bundling without incurring much additional processing overhead.
3. Allow a mix of agents, some of which support *Events* messages and some of which do not to interoperate within the same Zone and Context
4. Increase performance/scalability leveraging the asynchronous exchanges in the existing infrastructure as much as possible
5. Maintain the current levels of security & reliability of message delivery

Details of the Solution

Introduce a new SIF message type to bundle (concatenate) standard SIF Event Messages. On the HTTP/S transport this will be *SIF\_Events*. On the SOAP transport it will be *Events.*

In both cases the new message conforms to the existing SIF message structure, and contains its own Header, with elements defining the Message ID, Timestamp, and Source ID, security levels etc., relating to the *Events* message, and separate from the values associated with the individual Event messages it carries.

The *Events* message can best be thought of as a wrapper for one or more contained Event messages, each of which is character for character compatible with the XML that would define it if it was sent out alone on the wire. There is no attempt to optimize by reducing the duplication of header elements with identical values in each contained Event in the bundle.

However the contained Event messages in the *Events* message **will** correspond to the infrastructure stack expected by the recipient, no matter which transport they were originally sent out over. For HTTP/S transport recipients, this mandates that the SIF\_Message wrapping an *Events* message will contain internal SIF\_Message elements only surrounding a SIF\_Event, but no SOAP envelopes. On the SOAP transport, this means the SOAP Body of the *Events* will contain a series of SOAP Envelopes describing each internal Event, but will never contain any SIF\_Message elements.

**Issue #1: How will Event Messages be bundled for Pull Mode Agents?**

Currently the ZIS response to a Get\_Message, is a Status message with the complete returned message buried inside. There were two options for how to package an *Events* message in that response:

**1. Disallow Pull Mode Agents from receiving any *Events* messages**

This solves the problem by removing the use case. However it unfortunately reduces the effectiveness of the new *Events* message in addressing scalability issues in any Zone with one or more subscribing Pull Mode Agents. This was felt to be unacceptable.

**2. Bundle the Queued Event Messages and return an *Events* Message**

This approach relies on the fact that the “status” of all individual messages returned to a Pull Mode Agent in response to a GetMessage request to the ZIS, would be identical (i.e. “success”). If an error occurred in processing one of the Event messages for an Agent, it wouldn’t be queued, and only the previous successfully packed Event messages would be sent.

So with this approach, the Pull Mode Agent receives a SOAP Envelope containing the Status message returned in response to GetMessage. In this case, inside the SOAP Body of the Status message would be a SOAP Envelope containing an *Events* Message.

Inside the SOAP Body of the *Events* Message is a set of concatenated SOAP Envelopes, containing the Event Messages. In other words, an *Events* message is simply returned as the next message in the Agent’s Queue.

This is the solution that was chosen.

**Issue #2: Why not restrict Event bundling to only “homogeneous” Event messages?**

An earlier proposal to achieve Zone scalability at times of peak traffic was to bundle multiple Event Messages of the same event type (add, change, delete) and object type (ex: StudentPersonal) into a single “enhanced” Bundled Event message. Typically a Bundled Event was an Event containing changes to multiple objects of the same type. This addressed one of the driving use cases for Zone scalability (dealing with a flood of Attendance related Event messages at the end of a reporting period), and was probably a bit simpler to implement than the current approach.

There were two problems with this such Event Bundles:

1. **Not all Scalability needs were addressed**

There are several documented use cases where Zone throughput is impacted when a publishing agent changes a set of related objects, and does the same transaction over and over again. The result on the wire is a flood of published Event messages in the sequence Type 1, Type 2, Type 1, Type 2, repeated 1000’s of times.

The current *Events* message proposal handles this case while Event Bundling does not, since an Event Bundle cannot contain events of different types.

1. **Incremental Migration was more difficult**

The impact of the earlier proposal was almost solely limited to the Agents ... the ZIS just forwarded the new Bundled Events. The problem was that no publisher can be sure who is receiving its Events.

If the ZIS is not required to unpack Event Bundles, then an application cannot use them until each and every one of its subscribers can support them. This requires simultaneous upgrade to multiple applications and represents a significant barrier to technology adoption.

In addition, all subscribers and the publisher would have to be on the same underlying transport. Furthermore, it would be impossible to apply XML filtering to the Events contained in such a Bundle, because the ZIS was just routing the message without opening it.

In contrast, the *Events* message is primarily supported by the ZIS. As a result, publishers and subscribers can still communicate seamlessly, even in cases where one or more Agents support the *Events* message and the others do not.

Once it was clear that the ZIS needed to be involved in the solution for increasing Zone scalability, the *Events* message was determined to be the preferable technology choice.

Detailed Header Analysis

Even for Push Mode Agents, some concerns remain about supplying a top level *Events* message header which could potentially conflict with the header elements of one or more of the internal Event messages being bundled. This is addressed in more detail in the table below. Possible open issues are highlighted in bold.

|  |  |
| --- | --- |
| **Header Element** | **Potential conflict between *Events* Header element and one or more equivalent internal Event Header elements** |
| Infrastructure Version | Each internal Event message retains its Infrastructure version. It is possible that this version number is later than that indicated for the *Events* message which contains it.  This does not cause a conflict since all messages (including the *Events* message itself) were determined by the ZIS to be safe to send to this Agent |
| Data Model and Data Model Version | Potentially different values might exist between one or more of the internal Event messages, and / or with the *Events* message Header. But since no Event can be packed in an *Events* message which could not have been sent individually, there are no additional complexities introduced by the *Events* message. |
| Zone ID (SOAP Transport only) | Identical values for the *Events* message and the headers of all contained Event messages. No conflict. |
| Security Level | **An Agent sending an *Events* message may place different security requirements on each Event message within the *Events*. A**s the ZIS will break open the *Events* message and process each Event separately, each Event retains its security requirement and again there is no conflict.  As for ZIS generated *Events*, the wire must be secure enough to meet the highest requirement of all the internal Event messages. This is another reason the ZIS must create an *Events* message only from the set of Event messages already qualified as being deliverable to the recipient, and already in the recipient’s message queue. |
| Timestamp | No conflict. Internal Event messages retain their own timestamp of when they were issued. The *Events* message has a timestamp reflecting when the message was actually sent. |
| Message ID | No conflict. Each internal Event message retains its unique Message ID |
| Message Name (Web Service Action) | No conflict. Each internal Event message maintains its Name / Action.  “Events” must become a new message Name / Action value. |
| Destination ID | **An Events Message cannot be issued by an Agent with a Destination ID.** |
| Source ID | No conflict. Internal Event messages could have been originally issued by different applications. |
| Contexts | No conflict. An *Events* message may contain internal messages which have different contexts. |
| Packet Data | **An *Events* message cannot be packetized and Event messages do not support packetization.** |
| Special Header Elements | None known at this time. An *Events* message is not part of any message sequence (ex: Request / Response) and does not rely or depend on any other message, although it contains Event messages which have object type and Event type qualifiers..  Like all other SIF messages, the *Events* message delivery to Push Mode Agents is asynchronous and to Pull Mode Agents is synchronous (in response to a GetMessage) |

ZIS *Events* Message Processing

When the ZIS receives an *Events* message it will (assuming successful reception):

* Unbundle and persist the individual Event messages it contains
* Send an Ack with the matching MsgId of the *Events* (not of the individual Event messages)
* Process each Event message into the message queue of the appropriate Agent(s)

When the ZIS successfully adds an Event message to an Agent queue (assuming security levels and other factors are legal) it should check to see if there are already Event messages in the queue waiting to be Ack’d and whether the Agent in question supports reception of *Events* messages. If so, it may form a new “Events Group” or append the Event message to an existing “pending” *Events* message.

When either the maximum buffer size of the recipient would be exceeded, or a timeout period since the last message sent is reached, or a non-Event message is queued for transmission or a newly queued Event message cannot be simply added to the *Events* message and sent (ex: security level too high), the ZIS should close the Event group to new messages, and create a *Events* message and send it out to the recipient as soon as possible. Upon receiving the SIF\_Ack / ACK message the ZIS must remove all of the individual messages that were bundled in the *Events* message from the agents queue. If a SIF\_Error / Error is returned, the ZIS will still remove all of the individual Event messages from the queue, but will post a LogEntry Event describing the problem.

It should be noted that most such errors will be generated by network problems rather than the content of any individual Event message. Attempting to determine which Event messages in the *Events* message were “understood” and resending only the problem Event messages is considered to be counterproductive.

Therefore the rejection is all or nothing and no Event in the rejected *Events* message is resent to that subscriber.

Minimizing Latency

It is imperative that the effort to bundle Event messages does not introduce unnecessary latency that would defeat the main purpose of this proposal, which is to increase Zone performance and scalability.

For example, if the ZIS is receiving Event messages for an agent, and these qualify to be added to the *Events* message, the ZIS should do so, but only within a predefined time limit. In other words it is important that an algorithm similar to the Nagle algorithm be implemented.

Using such an algorithm, the ZIS would wait for the next Event message for the agent, hoping it qualifies to be bundled but would only wait a configurable amount of time. This would allow the ZIS to expedite the delivery of an *Events* message as soon as:

* A message is received that does not qualify to be part of the current *Events* message (for example if it is not an Event).
* The preset “maximum timeout before sending *Events* Message” has expired.

Bundle Processing

Upon receiving a *Events* message, an Agent must:

* Validate the *Events* header. If it is understood, issue a synchronous ACK. Otherwise issue a synchronous Error.
* Process each internal Event message from the top. If an error occurs, issue a LogEntry Event with the Message ID of the problem. Multiple LogEntry Events may result from reception of a single *Events* Message.

Upon receiving an *Events* message response (ACK or Error), a ZIS must:

* Discard all Event messages contained in the *Events* message from the Agent’s input Queue
* If an error was reported, publish a LogEntry Event
* Either create another *Events* message, starting with the next Event message in the Agent Queue, or send the next Event directly.

XML Examples

An *Events* message, with two embedded Event messages.

<?xml version="1.0" encoding="UTF-8"?>  
<SIF\_Message xmlns="http://www.sifinfo.org/infrastructure/2.x" Version="2.6">  
 <SIF\_Events>  
 <SIF\_Header>  
 <SIF\_MsgId>00000131E3FA956C000ACB57350F29AB</SIF\_MsgId>  
 <SIF\_Timestamp>2011-08-19T14:37:55-07:00</SIF\_Timestamp>  
 <SIF\_Security>  
 <SIF\_SecureChannel>  
 <SIF\_AuthenticationLevel>0</SIF\_AuthenticationLevel>  
 <SIF\_EncryptionLevel>0</SIF\_EncryptionLevel>  
 </SIF\_SecureChannel>  
 </SIF\_Security>  
 <SIF\_SourceId>SIF\_Empty\_Query\_Agent</SIF\_SourceId>  
 <SIF\_Contexts>  
 <SIF\_Context>SIF\_Default</SIF\_Context>  
 </SIF\_Contexts>  
 </SIF\_Header>  
 <SIF\_EventMessages>  
 <SIF\_Message xmlns="http://www.sifinfo.org/infrastructure/2.x" Version="2.6">  
 <SIF\_Event>  
 <SIF\_Header>  
 <SIF\_MsgId>00000131E3FEA016000ACB57350F29AB</SIF\_MsgId>  
 <SIF\_Timestamp>2011-08-19T14:42:20-07:00</SIF\_Timestamp>  
 <SIF\_Security>  
 <SIF\_SecureChannel>  
 <SIF\_AuthenticationLevel>0</SIF\_AuthenticationLevel>  
 <SIF\_EncryptionLevel>0</SIF\_EncryptionLevel>  
 </SIF\_SecureChannel>  
 </SIF\_Security>  
 <SIF\_SourceId>SIF\_Empty\_Query\_Agent</SIF\_SourceId>  
 <SIF\_Contexts>  
 <SIF\_Context>SIF\_Default</SIF\_Context>  
 </SIF\_Contexts>  
 </SIF\_Header>  
 <SIF\_ObjectData>  
 <SIF\_EventObject ObjectName="StudentPeriodAttendance" Action="Add">  
 <StudentPeriodAttendance   
 xmlns="http://www.sifinfo.org/infrastructure/2.x"  
 RefId="37E9CC9E655044BBF1D709C37139E1D2"  
 StudentPersonalRefId="25DA0E9DE36DFBC52616985D9638EA06"   
 SectionInfoRefId="287C8252A59456A29AE0E407F670DCB7"   
 SchoolInfoRefId="4D929BA335696BB101AE0C7241B1AFB0"   
 Date="2011-03-12">  
 <AttendanceCode   
 AttendanceCodeInfoRefId="0D4573AB0CD648E89BDB9130146F3204">  
 </AttendanceCode>  
 <AuditInfo>  
 <CreationUser Type="Other">  
 <UserId>d1P1HI</UserId>  
 </CreationUser>  
 <CreationDateTime>2011-07-22T00:39:55</CreationDateTime>  
 </AuditInfo>  
 </StudentPeriodAttendance>  
 </SIF\_EventObject>  
 </SIF\_ObjectData>  
 </SIF\_Event>  
 </SIF\_Message>  
 <SIF\_Message xmlns="http://www.sifinfo.org/infrastructure/2.x" Version="2.6">  
 <SIF\_Event>  
 <SIF\_Header>  
 <SIF\_MsgId>00000131E4018F4C000ACB57350F29AB</SIF\_MsgId>  
 <SIF\_Timestamp>2011-08-19T14:45:32-07:00</SIF\_Timestamp>  
 <SIF\_Security>  
 <SIF\_SecureChannel>  
 <SIF\_AuthenticationLevel>0</SIF\_AuthenticationLevel>  
 <SIF\_EncryptionLevel>0</SIF\_EncryptionLevel>  
 </SIF\_SecureChannel>  
 </SIF\_Security>  
 <SIF\_SourceId>SIF\_Empty\_Query\_Agent</SIF\_SourceId>  
 <SIF\_Contexts>  
 <SIF\_Context>SIF\_Default</SIF\_Context>  
 </SIF\_Contexts>  
 </SIF\_Header>  
 <SIF\_ObjectData>  
 <SIF\_EventObject ObjectName="StudentPeriodAttendance" Action="Add">  
 <StudentPeriodAttendance   
 xmlns="http://www.sifinfo.org/infrastructure/2.x"   
 RefId="37E9CC9E655044BBF1D709C37139E1D2"   
 StudentPersonalRefId="25DA0E9DE36DFBC52616985D9638EA07"   
 SectionInfoRefId="287C8252A59456A29AE0E407F670DCB7"   
 SchoolInfoRefId="4D929BA335696BB101AE0C7241B1AFB0"   
 Date="2011-03-12">  
 <AttendanceCode   
 AttendanceCodeInfoRefId=  
 "0D4573AB0CD648E89BDB9130146F3204">  
 </AttendanceCode>  
 <AuditInfo>  
 <CreationUser Type="Other">  
 <UserId>d1P1HI</UserId>  
 </CreationUser>  
 <CreationDateTime>  
 2011-07-22T00:39:55  
 </CreationDateTime>  
 </AuditInfo>  
 </StudentPeriodAttendance>  
 </SIF\_EventObject>  
 </SIF\_ObjectData>  
 </SIF\_Event>  
 </SIF\_Message>  
 </SIF\_EventMessages>  
 </SIF\_Events>  
</SIF\_Message>