



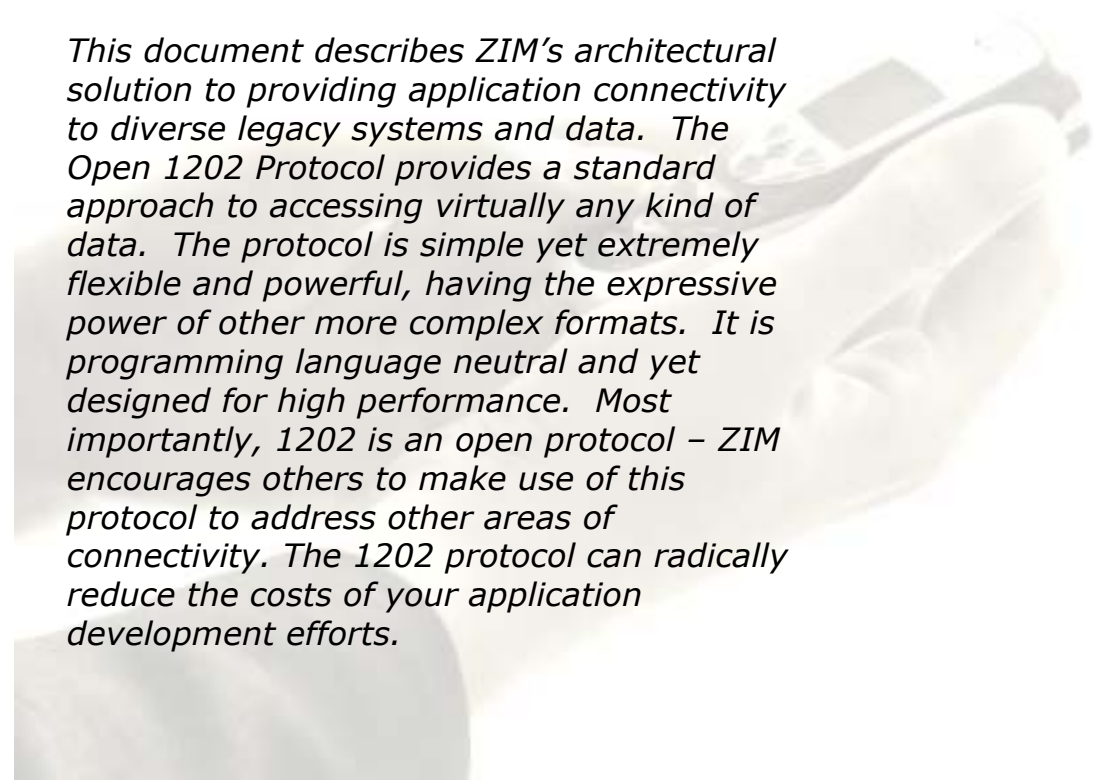
Unleashing the Power of SMS

The Open 1202 Protocol

A ZIM Whitepaper

**Version 01.00 July 2002
ZIM Technologies International Inc.**

This document describes ZIM's architectural solution to providing application connectivity to diverse legacy systems and data. The Open 1202 Protocol provides a standard approach to accessing virtually any kind of data. The protocol is simple yet extremely flexible and powerful, having the expressive power of other more complex formats. It is programming language neutral and yet designed for high performance. Most importantly, 1202 is an open protocol – ZIM encourages others to make use of this protocol to address other areas of connectivity. The 1202 protocol can radically reduce the costs of your application development efforts.

A faint, grayscale background image of a hand holding a mobile phone, positioned on the right side of the page. The hand is holding the phone in a way that the screen is visible, and the phone is slightly tilted. The image is semi-transparent, allowing the text to be read over it.



Unleashing the Power of SMS

The Open 1202 Protocol

Whitepaper

Version 01.00

July 2002

ZIM Technologies International Inc.

ZIM, Zim is Mobile, are trademarks or registered trademarks of ZIM Technologies International Inc.

Copyright © 2002 ZIM Technologies International Inc.



Contents

Executive Summary	4
The Open 1202 Protocol	5
System Development in the 21st Century.....	5
The ZIM Open 1202 Protocol	5
The Open1202 Protocol – An Architectural Approach	6
The 1202 Protocol In Practice	7
1202 and Zim 7	7
The Benefits of 1202 and the ZIM Application Connectivity Architecture.....	8
The 1202 Protocol – Technical Details	11
The Format of 1202 Messages	12
ZIM SMS Office – An Example Application of 1202	14
The Architecture of ZIM SMS Office	14
ZIM SMS Office is Implemented using the Open 1202 Protocol	15
Conclusion – 1202 is the Glue for Modern Application Development and Integration.....	19
About ZIM.....	20



Executive Summary

Modern application development often involves integration with existing systems (everything from custom software to packaged software). In particular, new software must be developed to access these existing systems. Solving this *connectivity problem* represents a significant portion of the development costs of new systems.

The Open 1202 Protocol developed by ZIM Technologies addresses the fundamental problem of connectivity head on and changes the way in which developers and systems integrators approach bringing disparate systems together. The protocol is simple yet extremely flexible and powerful. It provides data communications between components (known as *connectors*), which access existing systems and data, and the core application logic. It defines message formats that are composed of one or more fixed length segments (1202 bytes long – hence the name). Each segment is composed of seven fixed length fields that contain control information and data in plain text. 1202 has the expressive power of other more complex formats. It is programming language neutral and yet designed for high performance. Most importantly, 1202 is an open protocol – ZIM encourages others to make use of this protocol to address other areas of connectivity.

1202 delivers a number of major benefits in the context of modern application development:

- **1202** is powerful enough to handle all kinds of data formats and modes of communications but simple enough to deliver high application performance. The efficiency of communications between new application logic and existing systems is often a critical success factor for overall application performance. 1202 can handle the diverse needs of communication with many different kinds of systems, however, unlike more complex protocols, it can do so very efficiently.
- **1202** reduces development and maintenance costs by handling common communications requirements. 1202 can be used with any programming language and can be implemented on top of any transport protocol. Implementations of 1202 are available in most common programming languages and support all the common communications tasks. Development is simplified because programmers can now focus on the specific needs of their application rather than communication details.
- **1202** is an open protocol. Technical descriptions as well as implementations in a number of popular programming languages are available from ZIM. ZIM encourages others to create new connector components and can provide training and mentoring services as necessary.

ZIM Technologies has developed a large number of connectors that access many common systems such as SAP, Microsoft Office and Microsoft Exchange. The 1202 protocol has been used extensively in the implementation of ZIM SMS Office, an exciting new product from ZIM Technologies that enhances the Microsoft Office suite by extending Office functionality to over 600 million SMS-enabled devices around the world. Zim 7, the latest version of ZIM's popular integrated development environment, also offers extensive support for 1202.



The Open 1202 Protocol

System Development in the 21st Century

We have been building computer systems for over half a century now. One of the more obvious but often overlooked outcomes is that newer systems must be able to *talk* to older systems, whether they are relational databases, LDAP directories, packaged systems (for example, Microsoft Office or SAP) or custom software. Each external system has its own ways of permitting access. Each system has its own set of directives and parameters. Each system has its own specific data formats.

Of course, newer applications don't access older existing systems simply "because they are there". In fact, these systems hold the bulk of an organization's information and knowledge. Sometimes, this is just a fact of life (for example, the old accounts payable system written in Cobol that you haven't got around to replacing yet). In other cases, existing systems are specifically designed as repositories for certain kinds of information – the LDAP directory contains directory information about all people and systems within an organization, the human resources system is the single source for information about employees, and so on. Newer systems that need access to this information must do so through these existing systems.

In many cases, solving the *connectivity problem* represents the lion's share of application development (and maintenance) costs.

This whitepaper describes ZIM Technologies solution to the connectivity problem – the Open 1202 Protocol – referred to simply as 1202. The paper outlines the benefits of 1202, provides a technical description of 1202 and, in a final section, describes how 1202 forms the basis for ZIM's exciting new product, ZIM SMS Office.

This paper is designed to inform, educate and explain how 1202 can be used to improve any IT development environment and ultimately, improve the operations of any enterprise that relies on IT infrastructure for reaching mission critical goals related to the effectiveness of technology. By referring to this whitepaper and using the example given, it will be possible for developers or business managers to understand the implications of using 1202 within their development environments.

The ZIM Open 1202 Protocol

The Open 1202 Protocol developed by ZIM Technologies addresses the fundamental problem of connectivity head on and changes the way in which developers and systems integrators approach bringing disparate systems together. The protocol is simple yet extremely flexible and powerful. It is programming language neutral and yet designed for high performance. Most importantly, 1202 is an open protocol – ZIM encourages others to make use of this protocol to address other areas of connectivity.

1202 promotes a component approach to application development. It provides data communications between components (known as *connectors*), which access existing systems and data, and the core application logic. It defines message formats that are composed of one or more fixed length segments (1202 bytes long – hence the name). Each segment is composed of seven fixed length fields that contain control information and data in plain text. The control information is used to identify the name and type of specific pieces of data and also permits the segments to be arranged into arbitrary hierarchies. As a result, 1202 has the expressive power of other more complex formats such as XML but is much more efficiently processed.

1202 is a *wrapper protocol*. It organizes messages into segments but places no limits on the contents of each segment. 1202 is ideally suited to address the problems of modern application development:

- **1202** is powerful enough to handle all kinds of data formats and modes of communications but simple enough to deliver high application performance.
- **1202** reduces development and maintenance costs by handling common communications requirements. 1202 can be used with any programming language and can be implemented on top of any transport protocol.
- **1202** is an open protocol. Technical descriptions as well as implementations in a number of popular programming languages are freely available from ZIM.

The Open1202 Protocol – An Architectural Approach

1202 is deployed within a highly powerful and flexible application architecture, illustrated below, devised by ZIM to address application connectivity and integration.

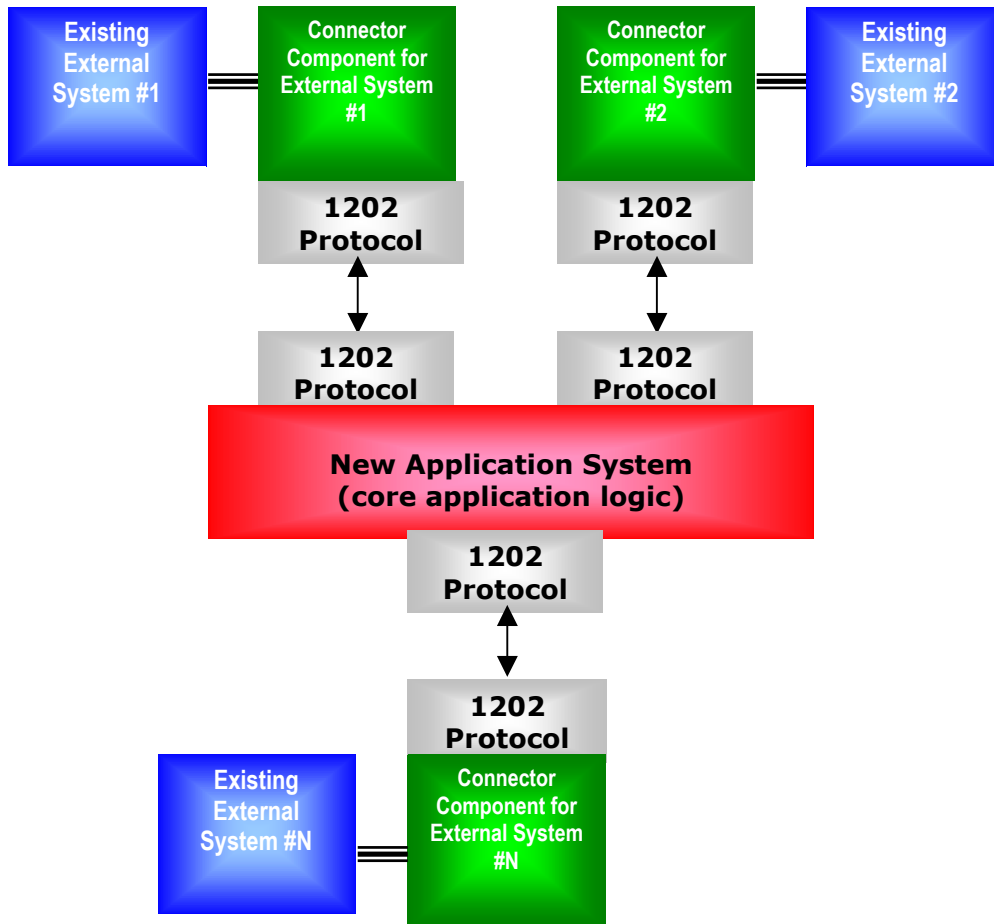


Figure 1 – ZIM Application Connectivity Architecture



The specific code for accessing any given existing external system (external system, for short) is encapsulated in a Connector Component as seen in Figure 1. For example, a connector component for a relational database could use SQL commands and the ODBC or JDBC API to access the database. Typically, but not always, the connector component will reside on the same hardware as the actual external system. The rest of the application logic passes information back and forth to the various connector components.

This architecture isolates the application logic from the software that accesses the existing systems. In this way, if some change occurs in the older systems, only the component accessing it needs to be changed – not the entire application.

The 1202 Protocol In Practice

ZIM has already applied 1202 to a wide range of applications including:

- Microsoft Exchange (see the detailed description of the Excon connector later in this whitepaper)
- Microsoft Outlook (calendars and contacts)
- Microsoft Excel
- LDAP (active directory)
- SAP
- Apache Servlet
- Zim 7 (ZIM's popular integrated development environment)
- GPRS and GSM modem connections
- SMTP
- JDBC
- XSLT
- Log File Reader
- Zim Application Monitor
- Command Line
- Interval Timer Service
- eSMTP

These applications of 1202 demonstrate the power and flexibility of 1202 in action.

1202 and Zim 7

In particular, Zim 7's support for 1202 (and enhancements planned for upcoming releases) will open up the world of Zim application development to a wide range of non-relational systems and data sources. Zim developers will be able to select best of breed components that can be combined to create a complete solution. Zim 7 currently provides the necessary infrastructure for using 1202 with Zim. Upcoming releases will include *connectivity classes* that will provide easy to use access (using 1202 as the underlying protocol) to many of the above types of external systems.

A more complete technical description of the 1202 protocol is provided later. However, at this point, let's examine the features and benefits of this architecture and 1202 in particular.



The Benefits of 1202 and the ZIM Application Connectivity Architecture

- **1202 supports communication between a connector component and the application initiated by the application, by the connector or by the external system.**

1202 supports both synchronous and asynchronous modes of communication. For example, event data sources can asynchronously produce messages that are sent (via 1202) to the core application logic. Synchronous conversations can be initiated by the application in order to request data, create transactions, etc. Interactions can be datagram or session oriented.

External systems come in all shapes and sizes. And the modes of interacting with them are equally varied. Some support a request-response style of interface. For example, relational database management systems provide a programmatic interface that allows requests to be submitted. Responses (data and/or exception information) are returned. For these kinds of systems, the application will typically initiate a request to the connector component. The connector will then interact with the external system and eventually send back a response to the application. With other kinds of external systems, the interaction may be initiated by those systems rather than the application. Consider the simple case of an external log file that the application wishes to monitor. In this case, the application will probably send an initial request to the connector component indicating it wishes to be informed of changes to the log file (the application *subscribes* to changes in the log file). The connector monitors the log file. When a log file event change occurs (for example, a record is appended to the file), the connector component initiates an interaction with the application. In other cases, the modes of interaction may be a combination of application-initiated and connector-initiated. Communication between the application and connector components needs to be flexible enough to handle all these modes of interaction. 1202 delivers that flexibility.

- **Communication between connector components and the application permits very efficient processing to maximize overall application performance.**

Because each 1202 segment and field is fixed length, there is no need for complex parsing algorithms. Fixed length formats also permit efficient communications using underlying protocols such as TCP/IP.

In most cases, overall application performance will depend on the performance of component communications. Typically, applications will access (or receive events from) external systems frequently. This means that the performance of the interactions between application and connector components can be a critical success factor for the application itself. Designs that minimize the costs of encoding and decoding of the data passing between the application and connectors will lead to optimal performance.



- **Connector components can be coded in a variety of languages.**

External systems may reside on different hardware platforms and may require or encourage access using specific programming languages. This means that connector components must be written in a wide variety of programming languages that happen to be best suited to the specific external system. A one-language-fits-all approach will not work.

The overall simplicity of 1202 (fixed format, plain text) means that it can be processed by virtually any programming language.

- **The 1202 protocol provides uniform support for communicating directives, external data, and events. This reduces overall costs of development and maintenance.**

Custom implementations of communications between the application and connectors are not a good approach. Although the content will vary from external system to external system, there are many design and implementation aspects shared by all systems. So, a much better approach is to design a wrapper protocol that handles the common aspects of all connector-application communications. As an analogy, consider the (now little practiced) art of letter writing. The contents of each letter are unique but they are all written on pages and enclosed in envelopes, both of which come in a number of standard sizes. This facilitates the writing of the letter and also makes its delivery more efficient. In the same way, a wrapper protocol will speed up the software development and maintenance process.

1202 is a wrapper protocol. It organizes messages into segments but places no limits on the contents of each segment. As a result, a segment may contain data values, external system directives, SQL commands, etc. It is up to the connector component and the core application logic to agree upon higher-level protocols that define the actual rules and meaning of the data. Since 1202 handles and simplifies the common communications tasks, development and maintenance costs are reduced.

- **The 1202 protocol is simple yet powerful. Development costs are reduced but not at the expense of functionality and performance.**

Simplicity will lead to high speed encoding and decoding of data and will permit the application to achieve its performance objectives. However, simplicity must not come at the expense of expressive power. The data that passes between external systems and the application will be in a wide variety of formats and its internal organization can be open ended. Sometimes the data will simply be a few specific data values, or it may be tabular in nature (for example, a result set from a relational database). Data may also be hierarchical (for example, data from an LDAP directory), or it may be organized in an even more complex fashion. Whatever the format and organization of the data, the wrapper protocol must be expressive enough to represent that data.

1202 uses fixed format segments and fields and all data is represented in plain text. It is easy to use and easy to debug, but the hierarchical structure of segments is semantically rich and allows for the representation of virtually any data structure. ZIM has already successfully applied 1202 to systems as diverse as LDAP directories, SAP, Microsoft Exchange, relational databases, and a number of custom systems. 1202 has clearly demonstrated its ability to handle any kind of external system.



- **1202 is an *open* protocol. ZIM makes 1202 software freely available. Application development and maintenance costs are further reduced.**

ZIM has created software to facilitate the creation, sending and reception of 1202 messages in a variety of languages including Java, C++, Visual Basic and Zim. By using this software, the programmer is relieved of having to know anything about the specific field format of segments or the details of the underlying communications protocol. Connector components and the application's core logic can be implemented much more quickly. Maintenance costs are also reduced because common software is used. There is less likelihood of errors and connector and application code is simplified.

ZIM is committed to maintaining 1202 as an open protocol. Complete technical descriptions of the protocol as well as detailed descriptions of its use in the many connector components already developed by ZIM are available. Software that assists in the creation sending and reception of 1202 messages is also available¹. ZIM encourages others to create new connector components and can provide training and mentoring services as necessary.

- **Connector components become reusable and replaceable components in the overall system.**

The ZIM Application Connectivity Architecture encourages the creation of true components that can be reused in other applications. In addition, these components can more easily be replaced (perhaps by an improved implementation) without affecting the rest of the application, as long as the new connector also uses the wrapper protocol and the content follows the same rules as well. This can lead to reduced development and maintenance costs.

A perfect example is the new ZIM SMS Office product from Zim Technologies that makes extensive use of 1202 and connector components². Some of these connector components access various components of the Microsoft Office product suite such as Excel. A systems integrator might choose to replace one of these connector components with one that provides better performance or additional functionality. This ability to extend the ZIM SMS Office product is possible as long as the new connector component provides the same 1202 interface to the rest of the system.

- **1202 speeds application development and maintenance because messages are all encoded as plain text.**

1202 messages are encoded in plain text. This eliminates common programming problems related to incorrect format (for example, a message recipient expected data in binary format but the sender provided it as an ASCII string) and byte order for binary data. The simplicity of the message format, combined with the standard protocol software, means that application coding is straight forward – accomplished more quickly and with fewer mistakes. Debugging is also eased by the simplicity of data formats. All these benefits apply to both initial implementations and ongoing maintenance.

¹ At this time, software is available in Java, C++, Visual Basic and Zim. Support for other languages will become available in the future. Check with your ZIM sales representative for the latest details on availability.

² SMS Office is described in more detail in a later section.



- **1202 can be implemented on top of any transport protocol.**

Current implementations use TCP/IP as the underlying transport protocol. SSL (Secure Sockets Layer) could be employed, in many cases at runtime. This would provide additional security for the messages passing between connector components and the application. Other implementations of 1202 could be easily developed that use other protocols such as HTTP.

The 1202 Protocol – Technical Details

This section provides a more detailed description of the Open 1202 Protocol from Zim Technologies. ZIM's 1202 protocol is a simple yet powerful messaging protocol developed to address the connectivity problem. It is typically used in combination with a connector component and is designed to provide a common interface for many different kinds of external systems and sources including:

Event Data Source

An event data source is an item that provides ongoing unscheduled information about a system. This information could come from an external process, a log file or a device. Event data sources are normally read-only and are monitored by a dedicated connector component. Event data that is received by the connector is converted from its native format and wrapped within the 1202 protocol before being sent.

Inquiry and Database Services

An inquiry service is a system that accepts requests and returns results. A connector component receives requests wrapped in the 1202 protocol and submits them to the external system. Results are then converted and wrapped in 1202 for return to the requestor. A database service is an inquiry service that also supports update operations. Requests may also provide support for defining transaction boundaries. These services may also provide transformational functionality (converting input data to another form, for example) or control functionality (the request is actually a directive to the external system to start, stop, etc.). In many cases, a simple request produces a response. However, in some cases, there may be no response to a request or a conversation is required before a final response is produced.

Publish/Subscribe Services

Publish/subscribe services accept information that is stored and is later transmitted to processes that have subscribed to it. Connector components receive publish or subscribe requests wrapped in 1202 and pass them on to the publish/subscribe service. Information to be sent to subscribers is intercepted by the connector and converted and wrapped in 1202.



The Format of 1202 Messages

A 1202 message is a simple position sensitive fixed width record bound by a *begin message* and an *end message* indicator. There are no linefeeds or carriage returns separating the message components.

All messages begin with a *begin message* indicator. The indicator specifies the beginning of the message the protocol version and the segment line length. An example *begin message* indicator is

```
BEGIN_MESSAGE00000000001202
```

that specifies that this is a *begin message* indicator, the protocol version number is zero and the segment length is 1202.

All messages end with an *end message* indicator. This indicator takes the form

```
END_MESSAGE
```

Between the two indicators, a message will normally contain one or more *message segments*. Message segments are 1202 characters long. Each segment is composed of fixed length fields.

- **Segment Reference Number**
Indicates related groups (see Group ID below) within the message.
- **Group ID**
Indicates a group of related segments.
- **Segment ID**
Indicates a segment number within a group.
- **Segment Type**
This field indicates the type of segment. 1202 does not distinguish different types of segment. However, senders and receivers may agree on special semantics for different segment types.
- **Data Name**
This field provides a name for the data value specified in the next two fields. 1202 does not interpret these names. They may be used, by common agreement, by message senders and receivers.
- **Data Type**
This field denotes the data type for the data value specified in the next field. 1202 does not interpret these type names. They may be used, by common agreement, by message senders and receivers.
- **Data Value**
The value to be associated with the data name and type.

As an example, consider an XSL translator connector component. This connector receives messages that specify a source and destination file as well as the name of an XSL translation file. For added security, the connector requires a password so that it can only be used by authorized applications. After completing the transformation, the connector sends back a "prompt" message.



Unleashing the Power of SMS

Here is an example message that could be sent to the translator connector component. Each row represents a segment (or the begin/end message indicator).

Segment Reference Number	Group ID	Segment ID	Segment Type	Data Name	Data Type	Data Value
BEGIN_MESSAGE0000000001202						
1	1	0	CONNECT	PASSWORD	CHAR	my_secret
1	2	0	TRANSFORM	SOURCE	CHAR	C:\source.xml
1	2	1	TRANSFORM	DESTINATION	CHAR	C:\transform.xsl
1	2	2	TRANSFORM	TEMPLATE	CHAR	C:\target.wml
END_MESSAGE						

This message provides the necessary password (my_secret, in this case), the name of the source and target file and the name of the XSL translation file. Once the translation is done, the connector component replies with the following message.

Segment Reference Number	Group ID	Segment ID	Segment Type	Data Name	Data Type	Data Value
BEGIN_MESSAGE0000000001202						
1	0	0	PROMPT			
END_MESSAGE						

Other connector components may use more complex messages and even require that a dialogue occur between the two ends of the communication channel before anything happens. There may be no response or the response may contain indications of various errors or exceptions that have occurred.



ZIM SMS Office – An Example Application of 1202

ZIM SMS Office is a new product from ZIM Technologies that enhances the Microsoft Office suite by extending Office functionality to over 600 million SMS-enabled³ devices around the world. ZIM SMS Office operates with Microsoft Outlook/Exchange, Excel, and Word. ZIM SMS Office is available in both an Enterprise Edition and a regular edition (known simply as ZIM SMS Office).

ZIM SMS Office enables the user to stay connected to their Office environment (including home offices) using cellular SMS devices, based on user-defined profiles. Notifications regarding Microsoft Outlook/Exchange meetings, received e-mails and changes to agendas can be pushed out to the user for information purposes or for immediate action. For example, a user can send and receive e-mails and can accept or reject an invitation to a meeting. ZIM SMS Office also supports Microsoft Word and Excel through the ability to populate tables and spreadsheets with real-time polling. Examples might include sales figures, inventory, or remote payroll entry.

The Architecture of ZIM SMS Office

ZIM SMS Office has been constructed using a highly modular architecture. Dozens of connector components co-operate (all using the 1202 protocol) to implement the product's functions.

While it is far beyond the scope of this white paper to describe the implementation of ZIM SMS Office in detail, a description of a key connector, the Exchange Connector, will illustrate the power of 1202.

The Exchange Connector (Excon for short) is the connector component that is used to receive events from a Microsoft Exchange server. Excon is a multi-threaded connector component that supports connections to many other processes. That is, it can publish event data to multiple targets and can receive control directives from many sources.

Excon operates on *feed lists* that are lists of Exchange users along with associated *Exchange message types*. Message types describe classes of events that may occur within Exchange (e.g. the reception of an e-mail by a user). A process may send a 1202 message to Excon subscribing to a specific feed list. The process will then receive 1202 messages in return every time an event occurs for an Exchange user and the user and message type are contained within the feed list. Control messages are sent to Excon to create, modify, and delete feed lists. These messages are acknowledged by a return message. A subscription message will cause Excon to send event messages as the events occur.

³ SMS (Short Message Service) is a standard protocol for sending messages to and receiving messages from cellular devices. It is supported on practically all of today's mobile devices.



As an example, let's first look at the messages sent to Excon by a process to set up a feed list.

Segment Reference Number	Group ID	Segment ID	Segment Type	Data Name	Data Type	Data Value
BEGIN_MESSAGE0000000001202						
MyRequest	1	0	COMMAND	DELETE	pass	MyList
MyRequest	2	0	COMMAND	ADD	pass	MyList IPM.NOTE Joe Blow
MyRequest	3	0	COMMAND	ADD	pass	MyList IPM.SCHEDULE. MEETING. REQUEST Joe Blow
END_MESSAGE						

The Data Type field must always contain the password for the Excon connector (for security reasons). In our example, Excon has been set up with a password "pass". The first message segment deletes a feed list called MyList. The second message adds an entry to MyList. Since it has just been deleted, the list is created. Then entries are added indicating that this feed list is interested in e-mail reception (IPM.NOTE) for user Joe Blow. The third segment adds an entry indicating the list is interested in meeting requests (IPM.SCHEDULE.MEETING.REQUEST) sent to Joe Blow. The strings such as IPM.NOTE are defined by Exchange to denote specific kinds of events.

Excon responds with a message indicating whether the request has succeeded or not. For example, if all goes well, the first message returned by Excon will be: (next diagram)



Unleashing the Power of SMS

Segment Reference Number	Group ID	Segment ID	Segment Type	Data Name	Data Type	Data Value
BEGIN_MESSAGE0000000001202						
MyRequest	1	0	PROCESSED	DELETE		MyList
MyRequest	2	0	PROCESSED	ADD		MyList IPM.NOTE Joe Blow
MyRequest	3	0	PROCESSED	ADD		MyList IPM.SCHEDULE. MEETING. REQUEST Joe Blow
END_MESSAGE						

Now a process may subscribe to the feed list called MyList by sending a message like this.

Segment Reference Number	Group ID	Segment ID	Segment Type	Data Name	Data Type	Data Value
BEGIN_MESSAGE0000000001202						
MyRequest	1	0	COMMAND	START	pass	MyList
END_MESSAGE						

If Joe Blow receives an e-mail, the process will receive the following message (because the process subscribed to MyList and we have indicated that MyList is interested in e-mail reception by Joe Blow).



Segment Reference Number	Group ID	Segment ID	Segment Type	Data Name	Data Type	Data Value
BEGIN_MESSAGE00000000001202						
MyRequest	0	1	DATA	MESSAGE_CLASS	CHAR	IPM.NOTE
MyRequest	0	2		LONGTERM_ENTRYID	CHAR	00000000DE712BF29D2A2748A9D5045EB1CAC224070023381869E7DC2D48B79FF1B2FDDFCDD1000000B4CDD00007E9632ACB647DC4BA6F9A3A7897DDF1F000000314C960000
MyRequest	0	3		RECEIVED_BY_NAME	CHAR	Joe Blow
MyRequest	0	4		IMPORTANCE	CHAR	1
MyRequest	0	5		SUBJECT	CHAR	The Sales Information You Requested
MyRequest	0	6		SENDER_NAME	CHAR	Bill Blow
MyRequest	0	7		SENDER_ADDRTYPE	CHAR	SMTP
MyRequest	0	8		SENDER_EMAIL	CHAR	bblow@zim.biz
MyRequest	0	9		PR_BODY	CHAR	Here are the figures you requested ...
END_MESSAGE						

Most of these entries are self-explanatory. It represents an e-mail sent by Bill Blow to Joe Blow and the message was received via an SMTP connection. The data in the second segment is the MAPI entry ID for this message (You may need this later if you want to access the same message).

As you can see from this example, it is very easy to set up feed lists and subscribe to them. Excon will then deliver lots of information about relevant events to you as they occur (in fact, Excon can react to approximately 30 different types of events). In the case of ZIM Office, this functionality is used to provide integration between Microsoft Outlook/Exchange and cellular devices through the use of SMS messaging. But this same information could be used to advantage in many other contexts.



Conclusion – 1202 is the Glue for Modern Application Development and Integration

Today, applications are rarely developed in complete isolation. They need access to data stored in and events produced by existing systems. Modern applications must integrate new software with these existing systems – a challenging task considering the diverse nature of these systems. A modular application architecture that employs a common data protocol between core application logic and the external systems is the key to successful and cost effective application development. ZIM's Open 1202 Protocol is ideally suited to operate within such an architecture. It provides language neutral support for the common communications problems faced by developers who are trying to integrate applications with existing systems. ZIM has successfully deployed 1202 within its own products, including ZIM SMS Office.

1202 is a simple yet powerful protocol capable of handling any mode of communication and any data format. Using ZIM's implementation of 1202, application development and maintenance costs can be greatly reduced.

We have gone further by making 1202 an open protocol so that others may enjoy the benefits of this technology. 1202 is a proven protocol that can be easily used to communicate with virtually any kind of external system or data source. It can be the glue for your future application development efforts.



Unleashing the Power of SMS

About ZIM

Zim is Mobile™

ZIM Technologies International Inc. is a leader in consumer and enterprise applications for the global SMS channel. ZIM brings mobile-enabling to a whole new level of responsiveness and action. ZIM is also an established provider of enterprise-class software and tools for designing, developing and manipulating database systems and applications. Through its SMS expertise and mobile-enabling technologies, ZIM bridges the gap between data and mobility.

ZIM is lead by Dr. Michael Cowpland, founder of Mitel Corporation (formerly, NYSE: MTL) and Corel Corporation (NASDAQ: CORL), and is headquartered in the high-tech epicentre of Canada - Ottawa, Ontario.

Contact Information

ZIM Technologies International Inc.
20 Colonnade Road
Suite 200
Ottawa, Ontario
Canada
K2E 7M6

Telephone: 613-727-1397
Fax: 613-727-9868

Email: info@zim.biz

www.zim.biz