First View™

Volume 4, Number 4 November 2004

Welcome to the Fall issue of ***First View***. Our objective, as always, is to keep you informed of developments at First Choice Software, and to recount what we've been up to since the last issue. This issue will highlight a number of new product releases planned in upcoming months, as well as describe a number of new license, subscription, and hosted service options for acquiring First Choice's innovative products.

This issue of ***First View*** discusses:

If you know of someone who might enjoy a copy of ***First View***, please share it with them. Also, have them send an email to add-firstview@fchoice.com. Make sure they include their name, phone number, and email address. We will send them future editions of the newsletter. If you do not wish to receive future copies of ***First View***, please send an email to remove-firstview@fchoice.com. Make sure your email address is either in the title or body of the email. We will remove you from our mailing list immediately.

From the CEO's Desk: Gaining Momentum as Clarify Customers Adopt First Choice

The first two quarters of 2004 resulted in a five-fold increase in revenues over the same two quarters in 2003, and as we wrapped up Q3 of 2004, year over year growth is showing continued significant growth. This momentum is attributable to one key factor: Clarify customers are adopting our web based solutions at an increasing rate as they discover the clear advantages First Choice offers over alternatives.

Based on customer feedback, we enjoy three distinctive key differentiators: Financial, Time (resources), and Functionality.

> Financially we deliver solutions that are typically well below alternatives in terms of license price, cost substantially less to maintain over the life of the investment, and may be purchased with flexible terms, several of which require no additional capital expense or increases to current budgets.
>
> We have taken great pains to deliver a highly flexible, easy to use, and productive customization environment. With our automated tools to accelerate migration of existing customizations, our customers are in production at a fraction of the time and effort of other alternatives. Combine this with a solution that works on any version of Clarify and database currently installed, and our customers have found there is simply no quicker way to implement a web-based solution.
>
> Our web-based products are built on top of **FCFL** (soon to be **FCFL.NET**) that enables us to deliver performance equal to or better than existing applications and with scalability to meet the most rigorous enterprise requirements. Additionally, our web-based applications incorporate value-added enhancements and extensions we developed over the past 9 years and that markedly improve a large percentage of the "Classic Client" implementations.

With the upcoming products described in this ***First View*** we fully expect customer adoption to continue to drive the momentum we have experienced over the last six quarters.

Technical Corner: Web Services for Clarify

In each issue of ***First View,*** the Technical Corner section offers tips, examples, and techniques we find useful. We include them, along with appropriate code examples, in the hope that you will find them useful as well.

Callable Objects for Clarify - History

For many years, we have heard many Clarify customers ask for a simple, generic, industry-standard mechanism to quickly make changes in Clarify – from ***ANY*** location, place, or program.

Certainly there have been partial solutions to this problem. The first, most common solution was to write direct SQL to the database. The problems with this approach are many and varied. The largest problem of all is that this is a very low-level solution, and that requires a great deal of custom programming and knowledge of the Clarify data model. While this solution can be used for anything, for most customers it is not an acceptable approach. Also, it is not always easy to call on direct SQL – some environments simply do not provide good SQL support.

Similarly, the C language API, eBusiness Framework (or CBOs), and ClearBasic from Clarify are simplistic solutions to the problem. And like direct SQL, they suffer from general applicability, and require too much manual coding and debugging.

Perhaps the most comprehensive solution previously written to solve this problem involved high-level APIs from First Choice. These APIs were written in both ClearBasic and as COM objects in First Choice's **FCFL** (First Choice Foundation Library) language. While these objects were easy to call from a COM/.NET environment, they were difficult to call from certain environments, including Java.

Callable Objects – from Anywhere

First Choice undertook the task of finding a good, general solution earlier this year. Our objective was to create a simple and quick mechanism that developers could use to query, insert, update, and delete data in Clarify, FROM ANY OTHER PROGRAM, INCLUDING ANY WEB PAGES ON ANY WEB SERVER!!! Not only that, but we wanted to make sure that both low-level and high-level operations were supported.

What would such an environment mean to you? It would mean that you could easily query data from Clarify, and display it, for example, on your web page. Later on, you could call on an API to perform an operation in Clarify (for example, create a case, modify a contact, install a site_part, anything). And all of this could be performed with a set of simple web service calls that anyone could insert into their program or web pages, and in minutes.

And while web services are fairly easy, in and of themselves, to call, we wanted to make the job even easier for those users calling our web services from either .NET or Java environments. For those users, we have created a *Web Services Client*. This client is a library of simple calls (either for Java or .NET), which greatly simplify calling the web services. It makes the web service calls trivial to make, presenting an **FCFL**-like and API-like interface that mimics what you would write in *fcClient*.

The goal has been accomplished. First Choice is now shipping version 1.0 of the *First Choice Web Services*. These services are SOAP-compliant, and can be called from virtually any web environment you may have. If you have a web server running, for example, on WebLogic, WebSphere, iPlanet, Tomcat, Apache, or IIS – you can have Clarify functionality running on your website!

The rest of this article presents an overview of the new, currently shipping *First Choice Web Services* product. Using this product, you can create programs, interfaces, customizations, and web pages that interact with Clarify in less time than you ever imagined possible.

A Web Service Architecture

The *First Choice Web Services* sit on top of several underlying components: The Clarify database, **FCFL**, and (optionally) the First Choice APIs. The *Web Services* is a discrete component that installs on a machine on the network. It does not have to be on a machine running **FCFL** and the database although we recommend it, but must be one that is running the IIS web server. After this, any other machine that can address the machine containing the *Web Services* via the HTTP protocol can access the *Web Services*.

For more detailed information about the architecture of the *First Choice Web Services*, please contact First Choice for the *Web Services u*ser guide.

Using the Web Services

The first step involved in using the *First Choice Web Services* is to install them. There are two installers you may choose to use: one for *Web Services*, and one for the web service client. The installers are automated. After you select a few details (installation directory, for example), all of the necessary software is installed. For more details on installation, see the *First Choice Web Services* installation guide.

Once the installation is completed, you will have the *First Choice Web Services* installed on your machine, and published as available on your network.

The next step of the process is, just use the *Web Services*. As mentioned above, you can access them simply by building up the proper XML message string, and passing it to the URL for the *Web Services*. This process is described, in detail, in the *Web Services* user guide.

Since most users, however, will be programming either with .NET or Java, it makes more sense to describe the process of accessing web services via the web service client. For the purposes of this article, we will show you example code written in C#, and executed in an ASP.NET web page. But the same process can be followed for Java-based programs.

To create the web service client object, you would place the following two lines of code in your web page:

```
fCWSClient fcWSClient = new FCWSClient();
fcWSClient.ResumeSessionOrLogin( sessionID, username, password );
```

The first line is responsible for declaring and creating the new web service client. The second line is responsible for logging in to the *Web Services*, or reconnecting to it, in case you have already created a connection to it.

When you are finished using the web service client, you can simply close it. For example:

```
fcWSClient.Logout();
```

That's it! Between the first two lines and the final logout line, you would place your web service calls to access Clarify.

Querying Data with a Web Service

It's very easy to query data using the web service client interface. In fact, for those of you who either have used ClearBasic or First Choice's **FCFL** language, it will look very familiar.

Simply put, you use ClearBasic-like constructs to set up and execute your queries. Primitives such as *TraverseFromParent*, *AppendFilter*, and *AppendSort* are all supported. The arguments are very similar to those you already know from **FCFL** and ClearBasic.

The following is a simple example of querying for a set of cases, and then finding the site and contact for each one. In reality you might choose to use a view for this purpose. But it was a quick, simple, and easily understood example for this article.

```
   FCWSGenericQueryClient caseGeneric = fcWSClient.CreateQueryGeneric("case");
caseGeneric.DataFields = "objid, id_number, title,
case_reporter2site,case_reporter2contact";
caseGeneric.AppendFilter("id_number", ">=", "55");
caseGeneric.AppendFilter("id_number", "<=", "66");
caseGeneric.AppendSort("id_number");

FCWSGenericQueryClient siteGeneric = caseGeneric.TraverseFromParent("site",
"case_reporter2site");
siteGeneric.DataFields =
"objid,site_id,name,status,cust_billaddr2address,cust_shipaddr2address";

FCWSGenericQueryClient contactGeneric =
caseGeneric.TraverseFromParent("contact", "case_reporter2contact");
contactGeneric.DataFields = "first_name,last_name";

DataSet ds = caseGeneric.Query();
```

The first section of code creates a new *generic* object. Generic objects (like in **FCFL**) represent one table or view in the Clarify system. In this situation, they represent the *case* table. The *DataFields* method allows you to decide which fields (and/or relations) to return for each row. The *AppendFilters* allow you to select which rows the query returns, and *AppendSorts* will perform sorting on the results.

The query is then set up to query a *child* generic object. Since we perform a *TraverseFromParent*, we will find the related row(s) in the child generic for each parent (case) generic found. We set the data fields so as to only return a few key fields for the site.

The code then does a second *TraverseFromParent* to get the contact for each case found.

Finally, the case generic is queried, and a *DataSet* is returned. When you query a generic object, you also automatically query any of the children generics underneath it. In this situation, all of the desired cases are found, as are the sites and contacts for each of those cases.

A few other important points need to be made with respect to querying generics that were not illustrated with the case above. First, the parent/child hierarchy can be as deep and complex as you need. You may perform a *TraverseFromParent* on any level of the query, as needed. You may query any relation cardinality (type) at all that you need. This includes OTO and MTM relations.

Also, you may use the concept of a *Bulk* to query more than one unrelated object at a time. You are not restricted to parent/child traversals only. For example, you could query for cases for a particular customer, as well as configuration items created last week all in one *Bulk*. Clearly these items are not related at all.

Once the query has been performed, an XML-format string will be returned with the results of the query. This XML can be traversed and the results displayed as you see fit. The following XML, for example, might be returned from the query above:

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <QueryResponse xmlns="http://www.fchoice.com/schemas/fcgenericws_v1">
      <QueryResult>
        <NewDataSet>
          <case objid="268435457"id_number="1" title="This is a test case1
           title."
                case_reporter2site="268435463"
           case_reporter2contact="268435466">
            <site objid="268435463" site_id="2" name="Henry's Site" status="0"
                cust_billaddr2address="268435463"
           cust_shipaddr2address="268435463" />
           <contact first_name="Henry" last_name="Smith" />
          </case>
          <case objid="268435458" id_number="2" title="This is a test case2
title."
           case_reporter2site="268435464" case_reporter2contact="268435464">
            <site objid="268435464" site_id="3" name="MacroSoft" status="0"
               cust_billaddr2address="268435464"
cust_shipaddr2address="268435464" />
            <contact first_name="Rae" last_name="Zor" />
          </case>
          <case objid="268435459" id_number="3" title="This is a test case3
title."
                case_reporter2site="268435462"
case_reporter2contact="268435462">
            <site objid="268435462" site_id="1" name="Joe's Site" status="0"
                  cust_billaddr2address="268435462"
cust_shipaddr2address="268435462" />
            <contact first_name="Joseph" last_name="HornBlower" />
          </case>
          <case objid="268435460" id_number="4" title="The Title for this Case"
                case_reporter2site="268435463"
case_reporter2contact="268435463">
            <site objid="268435463" site_id="2" name="Henry's Site" status="0"
                cust_billaddr2address="268435463"
cust_shipaddr2address="268435463" />
            <contact first_name="Marco" last_name="Polo" />
          </case>
        </NewDataSet>
      </QueryResult>
    </QueryResponse>
  </soap:Body>
</soap:Envelope>
```

Note that four cases were returned, and each one has one contact and one site. You can use any one of many tools currently available for parsing XML.

However, if you are using either .NET or Java (via the web service client), you do not even have to think about the XML. There is a much simpler way to proceed. If you are using the client, the XML returned is automatically placed

into a data structure for you. In .NET it is put back into a *DataSet* object, which you can traverse with standard ADO.NET primitives such as *ForEach.*

High-Level Operations with a Web Service

First Choice Web Services truly shine when it comes to complex, high-level operations. All of the **FCFL** API toolkits (containing over 500 different Clarify operations) are available to you as web service calls. Virtually anything you can do in the thick client GUI (or in First Choice's *fcClient* thin client) can be performed via *Web Services.*

Again, using the .NET client format, the following code (after the web service is created) will create a new case in the Clarify system:

```
result = this.fcWSClient.FCCSToolKit.create_case(
       siteID.Text,
       firstName.Text,
       lastName.Text,
       phoneNumber.Text,
       "", "", "", "", 0, "",
       title.Text,
       caseType.SelectedValue,
       "", "", "",
       phoneLog.Text,
       "",
       DateTime.Now.ToString(),
       DateTime.Now.ToString(),
       "", false, "", 0, "", 0, "", "", "", "", "", "");
```

It is important to understand that the *Web Service* APIs have ***exactly*** the same arguments and calling semantics as the **FCFL**-based APIs. These are full-featured APIs that can perform all of the required operations (all records, fields, relations are properly created, updated, and set). They validate all required data (returning architected return codes if anything is not correct), and perform all of their work in a single unit of work (to maintain data integrity).

The web service environment at First Choice contains a code generator that actually reads the APIs from the source files, and automatically generates the web service interfaces that you will call. This means that if APIs are modified, or new APIs are added, it is trivial to update the interface, and call on these APIs as part of *Web Services*.

In any event, it is very easy to be able to call on the high-level APIs and use them. Virtually any operation you might perform (Support, Quality, Logistics, Sales, Contract, Depot Repair, or Field Operations) is available. In addition, many administrative functions (both Policies & Customers Operations, as well as Product Manager) are also provided. For example, you could augment your "Create Employee" web page to also create the employee as a user in the Clarify system – all with fewer than 10 lines of code!!

Inserting/Updating/Deleting Data with a Web Service

It is also possible to insert, delete, update, and relate data rows using the *First Choice Web Services*. While this is not a very common operation (it's much easier to use a high-level API!), it is sometimes a requirement when you have data that does not map to an existing API.

As with querying, there are two ways to do this. One is to construct an XML string with the proper tags and attributes. Using the XML, you can perform these low-level operations. The string looks a lot like Data Exchange or **DIET** files.

However, like the querying example above, you can also use the web service client to easily insert, update, delete, and relate records. For example, the following web service code creates new contact, contact_role, site, and address records, and relates the properly. Remember, this can be performed from virtually any environment!!

```
// Create a new FCWSGenericUpdateClient to build update query
FCWSGenericUpdateClient generic = fcWSClient.CreateUpdateQueryGeneric();

// Create modify items for the tables we want to insert new records
InsertItem contactInsert = generic.InsertRecord("contactInsert", "contact");
```

```
InsertItem contactRoleInsert = generic.InsertRecord("contactRoleInsert",
"contact_role");
InsertItem siteInsert = generic.InsertRecord("siteInsert", "site");
InsertItem addressInsert = generic.InsertRecord("addressInsert", "address");

// Create a item for referencing an existing record
ReferenceItem stateProvRef = generic.ReferenceRecord("stateProvRef",
"state_prov");

// Append fields that we want to insert values into
contactInsert.SetField("first_name", firstName);
contactInsert.SetField("last_name", lastName);
contactInsert.SetField("phone", phoneNumber);

// Relate this inserted record to another ModifyItem
contactInsert.RelateRecords("contact2contact_role", contactRoleInsert);

// Append fields that we want to insert values into
contactRoleInsert.SetField("role_name", "Default");

// Relate this inserted record to another ModifyItem
contactRoleInsert.RelateRecords("contact_role2site", siteInsert);

// Get next Site ID from web service
string siteID = fcWSClient.FCSession.GetNextNumScheme("Site ID");

// Append fields that we want to insert values into
siteInsert.SetField("site_id",  siteID);
siteInsert.SetField("name", siteName);

// Relate the site to the address record
siteInsert.RelateRecords("cust_primaddr2address", addressInsert);
siteInsert.RelateRecords("cust_billaddr2address", addressInsert);
siteInsert.RelateRecords("cust_shipaddr2address", addressInsert);

// Append fields that we want to insert values into
addressInsert.SetField("address", address1);
addressInsert.SetField("address_2", address2);
addressInsert.SetField("city", city);
addressInsert.SetField("state", state);
addressInsert.SetField("zipcode", zip);

// Relate this new address to an existing state record
addressInsert.RelateRecords("address2state_prov", stateProvRef);

// Select an existing record to use as a relation for addressInsert
stateProvRef.AppendUniqueFilter("full_name", state);

// Run the query and retrieve the results
FCGenericModifyResult[] results =  generic.Update();
```

The first line creates a *GenericUpdateClient* object. This object is used to hold the records for insert and update. The next section declares four *InsertItem* objects – each one is a data row to insert. Then, a *ReferenceItem* is declared. This represents a row already existing in the database – it is much like a *Reference* in Dataex syntax.

The next section sets some data fields. This is followed by a relating of the contact to contact_role records. More data is inserted and related, and then the *FCSession.GetNextNumScheme* method is called to get the site_id field. Remember that all useful methods of **FCFL** are made available to you with *Web Services*. The site and address records are populated, and then the overall *GenericUpdateClient* object is updated, which causes all of the records to be inserted and related properly in a bulk.

Putting it all Together

This article provides a very quick introduction to the *First Choice Web Services*. Even with the brief nature of this article we are sure that you can see the power and flexibility of the *Web Services*. You can integrate them into any program or environment you wish. While they are particularly powerful when called from web pages, they can also be added to command-line or LAN-based applications. In truth, what you can do with them is virtually unlimited

Product News: Fall Releases Greatly Expand FCS Solutions

We were tempted to dedicate this entire issue to the recent set of product releases from First Choice. Certainly, we have written and shipped a great many in the past few months. We believe that these products will greatly assist Clarify customers in improving their environments and daily work. This section provides a brief highlight on what is currently available and use upcoming issues to provide more detail on each product. In the meantime, please contact us at sales@fchoice.com to learn more.

Expanded Application and fcClient Support

We have recently released *fcClient* 3.5, which offers substantial new functionality and performance. In the application area, we have added new logistics capabilities to Parts Requests for *Depot Repair* and *Spares Manager*. Pick, Fulfill, Ship, and Receive actions are now supported in the FCS thin client as well as the ability to create and update depot repair records, including adding labor and material entries and ECO's.

Additionally, we have added the underlying API toolkits for Field Operations and Depot Repair as well as upgraded the API toolkits for ClearContracts, ClearQuality, ClearSupport, and Interfaces to address a number of minor enhancements.

In response to customer requests for a cleaner Windows environment for *fcClient* we have also included a new Windows Manager for *fcClient* that allows any open window to be viewed and fronted from the console as well as the option to close all open windows (automatically when logging out). *fcClient* 3.5 contains over one hundred enhancements to improve your daily work – download the user guide for more details.

State of the Art Integration of Clarify with Other Applications

The ability to interface with Clarify has been significantly enhanced with the release of *First Choice's Web Services and Dot Net Wrappers* (fcws 1.0). The Technical Corner article found in this edition of ***First View*** provides more details in the applicability and usage of these powerful services.

First Technical Preview of .NET Architecture Now Available

The first two releases of our **FCFL.NET** architecture (technical preview and beta) are now available for **FCFL.NET** subscription customers and selected partners. This release is targeted towards customers and partners who wish to exercise the First Choice architecture through custom .NET applications. The GA release of **FCFL.NET** (due within the next month) will include a compatibility layer permitting existing applications and *fcClient*s to utilize **FCFL.NET** technology without modifications.

Upcoming 2004 Releases

The Following are Some of the Releases We are Working On for the End of 2004, and for the First Two Weeks of 2005:

Version 1.0 of **FCFL.NET** will be released late in 2004. **FCFL.NET** is a "from the ground up" implementation of our superior **FCFL** environment, written in C# for the .NET world. **FCFL.NET** is object-oriented, allowing for easy subclassing and customization. It is multi-threaded, making **FCFL.NET** significantly more scalable, and better performing than **FCFL** (which is already the fastest solution available to customers today).

The "Compatibility Layer" allows existing **FCFL**-based applications (including *fcClient*) to work with **FCFL.NET** without having to change a single web form!! Hundreds of other improvements (a license manager and an improved logging manager, to name just two) will make this the development environment of choice for all Clarify customers. Contact us at sales@fchoice.com for more details about getting a copy of **FCFL.NET**

First Choice will soon ship the newest version of our newest API toolkit: *FCFL API Toolkit for ClearSales*. This toolkit allows customers to add Sales functionality (Quote, Action Item, Opportunity, Lead, and more) to their applications quickly and easily.

Later this year, First Choice will add the first *fcClient* functionality in the Sales area. The first piece of functionality to be implemented will be Action Items. These will be full-featured Action Items that contain all of the bells and whistles you have come to expect from First Choice. They will also integrate fully with the console, Recent Objects, fcQuery, and more. Towards the end of the year, First Choice will also release a copy of *fcClient* with our new Account Management functionality.

fcAdmin is being greatly expanded. Soon you will be able to administer all of the following graphically from *fcAdmin* (in addition to the many GUIs available to you today): Geography (Countries, States, Time Zones, Currencies, and Zipcodes), Lists (Status Codes and Clarify Lists), ClearLogistics Transitions, Configuration Items, Parts (Part Domains, Part Numbers, Mod Levels, and more).

Employee Profile: Chad Levert

We are extremely pleased to welcome Chad to First Choice. He'll be managing our many implementation and professional services engagements. Even though our environment is very straightforward to implement, especially with the tools we have made available to assist in migration of existing customizations, there is typically a short implementation process to complete. Chad's role is to oversee these implementation processes and manage the appropriate resources (internal resources and external contractors) to ensure a successful customer experience.

Prior to joining First Choice, Chad served in several different capacities at Dell Computers, from web programmer and project manager to systems analyst, architect, and development manager. The last two years at Dell he served as the development manager for the high traffic Electronics and Accessories website (http://accessories.us.dell.com/sna/default.aspx?c=us&l=en&cs=19). During that time Chad was responsible for delivering several large projects including a merge of the Premier and e-commerce websites, and two COM/.NET conversions.

Chad served in the 10th mountain infantry unit during the Gulf War, and then went on to graduate from Southwest Texas State University, studying Computer Science and Business Administration. After graduation he had a brief stint at the US Department of Veteran Affairs as an application programmer before joining Dell.

Personal interests include riding motocross, softball, and competing in the Texas Water Safari (a three day, 260 mile canoe race).

Sales and Marketing News: New Options for Acquiring First Choice Products

Along with the many new products we've just released, we have developed a number of new licensing options. These options are based on customer feedback concerning the challenge they face of having to move their applications forward while still working under constrained Capital Investment budgets.

The First Choice product development philosophy has always been to build products that products offer customers tremendous flexibility in meeting their application needs. Offering customers similar flexibility in licensing only makes sense; so we are pleased to announce a variety of licensing options.

First, we continue to offer traditional enterprise licensing based on concurrent connections and installed thin client applications. With the implementation of our .NET architecture, licensing is extremely easy to manage, highly tailored, and based on need. In keeping with our value based pricing objectives we have maintained price points well below alternative solutions. Enterprise licensing is typically considered a capital expenditure and is accompanied by an annual support and maintenance agreement.

Second, we now offer a modified enterprise-licensing model. This is an interesting option that is particularly attractive to customers who have a reduced dependency on the classic client and who are moving to a thin client environment. First Choice products operate on any version of Clarify and virtually any database system. Further we can provide Clarify Help Desk support as part of our offering.

Third, we anticipate an aggressive roll-out of products based on our .NET architecture, including a complete Amdocs replacement that includes a broad suite of clients, data model, administration tools, and business process services - e.g. Rules Manager alternative and advanced email management. We expect the new product suite to be rapidly adopted by existing Clarify users who want more flexibility, greater performance and scalability, and lower cost of ownership. Customers have already asked us how to license these alternatives when available. To make the transition as simple as possible we are offering a subscription option: that for an annual fee subscription customers can receive the .NET based products and technologies necessary to implement First Choice's advanced .NET platform, or even a complete a migration to a 100% First Choice environment.

Finally, our new products and architecture provide a state of the art customer support solution that will be attractive to both new (non-Clarify) customers as well as existing Clarify users. Industry research reveals a significant shift is occurring in the CRM market towards solutions delivered as Internet-based services. By licensing applications as a service, rather than as a traditional enterprise license, customers can lower the cost of entry, reduce IT expense, and more rapidly adapt to changing needs. In the near future First Choice will offer our thin client solutions as a hosted service, as well as premise based solution, with pricing based on a straightforward cost per agent/per month. With no dependence upon Clarify components this option will be applicable to any company looking for highly effective customer support solutions.

Now, more than ever, acquiring First Choice solutions couldn't be easier.