

WebFOCUS

Security and Administration
Version 4 Release 3.6

Preface

This documentation is intended to help administrators determine how to deploy security and administration options in a WebFOCUS environment.

How This Manual Is Organized

This manual includes the following chapters and appendixes.

Chapter/Appendix		Contents
1	<i>Security Overview</i>	Presents a general overview of security and provides details about various types of security control.
2	<i>Providing Data Source Security: DBA</i>	Describes how you can use DBA security features to provide security for any WebFOCUS data source.
3	<i>WebFOCUS Security</i>	Provides information about securing the WebFOCUS Reporting Server, WebFOCUS Client, and Managed Reporting Environment (MRE).
4	<i>WebFOCUS Custom Security Exits</i>	Describes and provides examples of user written security exits.
5	<i>WebFOCUS Administration</i>	Describes the primary WebFOCUS Client configuration files, and their associated settings. It also contains information about WebFOCUS Script (WFS) commands and how to use them.
6	<i>Tips and Techniques</i>	Provides WebFOCUS security tips and techniques, such as how to create WebFOCUS site profiles, and how to use WebFOCUS profile application security.

Documentation Conventions

The following conventions apply throughout this manual:

Convention	Description
THIS TYPEFACE OR this typeface	Denotes syntax that you must enter exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable) in syntax for a value that you or the system must supply.
<u>underscore</u>	Indicates a default setting.
<i>this typeface</i>	Represents a placeholder (or variable) in a text paragraph, a cross-reference, or an important term.
this typeface	Highlights a file name or command in a text paragraph that must be lowercase.
this typeface	Indicates a button, menu item, or dialog box option you can click or select.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices; type one of them, not the braces.
[]	Indicates a group of optional parameters. None are required, but you may select one of them. Type only the parameter in the brackets, not the brackets.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis points (...).
. . .	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our World Wide Web site, <http://www.informationbuilders.com>, to view a current listing of our publications and to place an order. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about WebFOCUS security and administration?

Call Information Builders Customer Support Service (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 a.m. and 8:00 p.m. EST to address all your WebFOCUS questions. Information Builders consultants can also give you general guidance regarding product capabilities and documentation. Please be ready to provide your six-digit site code number (*xxxx.xx*) when you call.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our World Wide Web site, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answer to frequently asked questions.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Information You Should Have

To help our consultants answer your questions most effectively, be ready to provide the following information when you call:

- Your six-digit site code number (*xxxx.xx*).
- Your WebFOCUS configuration:
 - The front-end you are using, including vendor and release.
 - The communications protocol (for example, TCP/IP or HLLAPI), including vendor and release.
 - The software release.
 - The server you are accessing, including release (for example, 4.3.1).
- The stored procedure (preferably with line numbers) or FOCUS commands being used in server access.
- The name of the Master File and Access File.

- The exact nature of the problem:
 - Are the results or the format incorrect; are the text or calculations missing or misplaced?
 - The error message and return code, if applicable.
 - Is this related to any other problem?
- Has the procedure or query ever worked in its present form? Has it been changed recently? How often does the problem occur?
- What release of the operating system are you using? Has it, WebFOCUS, your security system, communications protocol, or front-end software changed?
- Is this problem reproducible? If so, how?
- Have you tried to reproduce your problem in the simplest form possible? For example, if you are having problems joining two data sources, have you tried executing a query containing just the code to access the data source?
- Do you have a trace file?
- How is the problem affecting your business? Is it halting development or production? Do you just have questions about functionality or documentation?

User Feedback

In an effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual. Please use the Reader Comments form at the end of this manual to relay suggestions for improving the publication or to alert us to corrections. You can also use the Document Enhancement Request Form on our Web site, <http://www.informationbuilders.com>.

Thank you, in advance, for your comments.

Information Builders Consulting and Training

Interested in training? Information Builders Education Department offers a wide variety of training courses for this and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our World Wide Web site (<http://www.informationbuilders.com>) or call (800) 969-INFO to speak to an Education Representative.

Contents

- 1 Security Overview1-1**
 - Elements of Security..... 1-2
 - Authentication 1-2
 - Access Control 1-3
 - Confidentiality..... 1-3
 - Data Integrity..... 1-3
 - Security Policy 1-4
 - Operating System Security 1-4
 - Network Layer Security 1-5
 - Web Server Security..... 1-6
 - Web Server Authentication 1-6
 - Web Server Environment Variables 1-7
 - Web Server Access Control Using Aliases 1-7
 - Web Server Confidentiality and Data Integrity 1-8
 - Database Security 1-8

- 2 Providing Data Source Security: DBA2-1**
 - Introduction to Data Source Security 2-2
 - Implementing Data Source Security 2-3
 - Identifying the DBA: The DBA Attribute 2-5
 - Including the DBA Attribute in a HOLD File 2-6
 - Identifying Users With Access Rights: The USER Attribute 2-6
 - Establishing User Identity 2-7
 - Specifying an Access Type: The ACCESS Attribute 2-9
 - Types of Access 2-10
 - Limiting Data Source Access: The RESTRICT Attribute 2-12
 - Restricting Access to a Field or a Segment 2-13
 - Restricting Access to a Value..... 2-15
 - Placing Security Information in a Central Master File 2-17
 - File Naming Requirements for DBAFILE 2-19
 - Connection to an Existing DBA System With DBAFILE..... 2-19
 - Combining Applications With DBAFILE..... 2-20
 - Summary of Security Attributes..... 2-21
 - Hiding Restriction Rules: The ENCRYPT Command..... 2-22
 - Encrypting Data..... 2-22
 - Performance Considerations for Encrypted Data 2-23
 - Restricting an Existing FOCUS Data Source 2-23
 - Setting a Password Externally 2-24

WebFOCUS FOCEXEC Security	2-24
Encrypting and Decrypting a FOCEXEC.....	2-24
3 WebFOCUS Security	3-1
WebFOCUS Reporting Server Security	3-2
WebFOCUS Reporting Server Security Settings	3-2
WebFOCUS Reporting Server Deployment Options	3-4
WebFOCUS Reporting Server Processing Modes	3-4
Pre-Started Agents.....	3-5
Windows NT/2000 Domain Security Considerations	3-6
WebFOCUS Reporting Server Administrator ID.....	3-6
WebFOCUS Client Security.....	3-6
Connecting to a Secure WebFOCUS Reporting Server	3-7
Setting WebFOCUS Security Defaults.....	3-7
WebFOCUS Signon Processing	3-8
WebFOCUS Cookies	3-9
Controlling Cookie Expiration	3-11
Encrypting the WebFOCUS User Cookie	3-11
Integrating Web Server Security With WebFOCUS	3-12
WebFOCUS and Network Layer Security	3-12
WebFOCUS and Data Source Security Integration.....	3-13
Managed Reporting Environment (MRE) Security	3-13
Built-In Security Features	3-13
MRE Signon Processing.....	3-14
MRE Signon Integration	3-16
MRE Internal Variables.....	3-17
Securing the MRE Repository.....	3-19
Deferred Receipt Security	3-19
ReportCaster Security.....	3-21

4	WebFOCUS Custom Security Exits.....	4-1
	WebFOCUS CGI Callable Exits	4-2
	WebFOCUS Reporting Server Input Processing (TRANSIN).....	4-3
	WebFOCUS Reporting Server Output Processing (TRANSOUT)	4-4
	Retfuncs.....	4-5
	Compiling a WebFOCUS CGI Callable Exit	4-12
	WebFOCUS Servlet Callable Exits	4-12
	MRE Callable Exits.....	4-16
	MRCEXT Header File for MRE Callable Exits	4-17
	MRCEXT0 Exit.....	4-20
	MRCEXT1 Exit.....	4-23
	MRCEXT2 Exit.....	4-25
	MRCEXT3 Exit.....	4-29
	MRCEXT4 Exit.....	4-30
	MRCEXT5 Exit.....	4-32
	Compiling MRE Callable Exits.....	4-37
	WebFOCUS DBA Exits	4-38
	Pre-Verify User ID Exit.....	4-43
5	WebFOCUS Administration	5-1
	WebFOCUS Client Configuration Files.....	5-2
	Encrypting WFS Files	5-6
	IBIWEB.CFG Settings	5-7
	MR_PGM_FLAGS	5-8
	IBIDIR.WFS Settings.....	5-9
	SITE.WFS Settings.....	5-11
	Web Server Settings	5-12
	WebFOCUS Script (WFS) Commands	5-13
6	Tips and Techniques.....	6-1
	Creating a WebFOCUS Site Profile	6-2
	Recommendations for Securing a WebFOCUS Web Site.....	6-3
	Minimum Recommended NTFS Permissions	6-4
	Application Security Integration	6-6
	Ensuring that the WebFOCUS Logon Is Invoked	6-8
	Index	I-1

CHAPTER 1

Security Overview

Topics:

- Elements of Security
- Security Policy
- Operating System Security
- Network Layer Security
- Web Server Security
- Database Security

Security concerns the protection of data against unauthorized access. However, even in the context of information systems, people have different ideas about what security means. To some, security centers on logon IDs and how to make passwords inviolable. To others, security relates to file and directory permissions. These ideas are all related to the concept of information systems security, but alone do not define it.

Your security system includes operating system security along with network layer security. In some cases you may be concerned with authenticating users, but not data integrity once those users have been allowed to access an application. For details about security options, see *Elements of Security* on page 1-2, and topics related to the specific types of security you may need to apply.

Elements of Security

There are four elements of security:

- Authentication
- Access Control
- Confidentiality
- Data Integrity

Authentication

The most fundamental element of security is identifying and authenticating a user's credentials. It is important to establish a systematic method for proving the identity between two or more entities, which are usually characterized as users and hosts. Without certainty that users are who they claim they are, a sophisticated security system is not possible.

Organizations need to decide the degree to which authentication is important in their applications. Frequently, different requirements for authentication will exist at different points within an application. There are many ways to support the needs of an organization, but it is important to consider the following early in the planning phase:

Defining the Importance of Authentication. For some applications, authentication may not be critical. However, when dealing with sensitive information, stricter authentication measures must be used. For example, allowing manager access to sales forecast data would require certainty about each manager's identity.

Determining the Method of Authentication. A second issue to consider is the means by which you will authenticate the user. There are both technical and policy aspects to this issue. The technical aspect concerns what passwords or personal identification numbers (PINs) to use, while policy concerns aspects such as when these codes will expire and how many characters or digits are allowed.

Establishing Where the Authentication Should Occur. A third consideration is the point at which the authentication will be performed. For example, authentication may take place when logging into the operating system, while a further layer of authentication might be required when accessing an application such as WebFOCUS. In another example, authentication may take place when logging on to the operating system, while the application then allows all already authenticated users to access resources without reauthenticating (this is called "trusted" authentication). It is likely that the decision of where authentication should occur will be made in conjunction with other elements of security, with special consideration being given to where access control will be enforced. The user's perspective should also be taken into account to minimize the number of passwords required for authentication.

Access Control

Once you have appropriately authenticated someone, the next step is to determine and then enforce an appropriate level of access.

Enforcing Access Control Privileges. Access control is typically thought of in the context of file systems, which can control read and write access to data on a disk. Some systems support only read and read/write control, while others offer specific privileges, such as the ability to modify an object's permissions. A robust method for enforcing access control privileges is by maintaining user and group level permission schemes.

Assigning Group Permissions. Deciding who receives what set of privileges typically involves assigning users to user groups, and then enforcing permissions at the group level. Group level security enables organizations to move users in and out of groups without requiring a change to the individual user's permissions.

Administering Access Control. A system administrator should be assigned to manage access control privileges. Automatic auditing of privilege changes may also be a worthwhile feature to consider.

Confidentiality

Another important element of security is confidentiality. Confidentiality ensures privacy using encryption. Various forms of encryption include data, network session, and file-based encryption. Encryption may be relatively weak or strong, and can be based on any number of private or public encryption schemes.

Like the other elements of security, a decision regarding which data is sensitive and how important it is to keep this data confidential is different for every organization.

Data Integrity

A primary concern about sensitive transaction-based applications is data integrity. Data integrity is the assurance that information can only be accessed or modified by those who have been given authorization. It is important that all data or metadata be unalterable except by those users who have been given privileges to make changes.

Security Policy

The decisions an organization makes about security issues should be formalized in a written policy document. Such a document serves as a reference point for parties with different points of view. For example, during the planning stage of a WebFOCUS implementation, a policy document may lead the design team to eliminate certain platforms from consideration. In addition, questions on how to establish application security can be predefined by an internal security group.

Part of an organizations's security policy should include methods for establishing physical barriers that control access to sensitive resources. For example, many computer rooms are now secured with an access card controlled door. A truly secure application must have its servers secured from being physically damaged.

A security infrastructure that already exists can be used as a starting point for a new security system, or a security policy can be created from scratch. Either way, having an awareness of the security policy can help to make an application more effective.

Operating System Security

Security measures are provided by an operating system and are enforced at one or both of the following locations:

- At the user's entry point into either a local or shared computing resource. Shared computing resources include both distributed (or networked) systems and centralized (mainframe or midrange) systems.
- On a system where application services are being requested.

Operating system security may integrate all four elements of security: authentication, access control, confidentiality, and data integrity. How these elements are implemented depends on the specifics of each platform and operating system. For more information, consult your operating system documentation.

Network Layer Security

Many organizations employ security measures within the network infrastructure of a distributed system, or at access points to centralized systems. Network level security segments the network and allows only authorized users with the appropriate permissions to access resources.

A common example of network level security is a **firewall**, which shields a trusted network from an untrusted network. A firewall protects the inside environment from the external environment, similar to the way a gate with a lock prevents people without the proper key from entering a secure area. A firewall can also prevent insiders from accessing the outside network. For example, during working hours all Web access might be restricted, but after working hours insiders are allowed access to the entire Web, thus eliminating Web surfing during the day. Firewalls can also restrict certain types of traffic on the network (for example, heavy traffic that may be difficult to manage).

Firewalls are usually a combination of a filter and a proxy gateway:

- A **filter** typically plays the role of a traffic cop, examining the header of each packet and applying a set of rules that decide whether the packet should be sent to its destination.
- **Proxy gateways** act as application servers to a client and a server. They are typically broken into two types:

Generic proxies are not application specific. A client that wants to connect to an outside server connects to the proxy server, and then the proxy server behaves on behalf of the client and connects to the outside server. The proxy server typically implements some type of authentication to validate whether a particular user request should be sent.

Application proxies have an inherent knowledge of the application protocol they are supporting. For example, an application proxy can be set up to allow an FTP application to bring files into an organization, while disabling the ability to FTP files outside the organization.

Web Server Security

Web server security requires all the key elements of security, including Web server:

- Authentication.
- Environment variables.
- Access control using aliases.
- Confidentiality and data integrity.

Web Server Authentication

Typically, Web servers provide several authentication options that can be configured by an administrator. It is important to understand how Web server authentication is performed, because many Web security problems can be traced to a misuse or misunderstanding of this setting.

The major authentication options are as follows:

- **Anonymous.** A special system account is established and all Web users act with the security profile of this account. No user IDs or passwords are exchanged and everyone has the same access to resources. Unless some other form of identification is requested, there is no way to distinguish between users.
- **Basic.** A logon ID and password are requested by the Web server for authentication. This information is transmitted unencrypted across the network. However, it is encrypted when basic authentication is used with the Secure Sockets Layer (SSL) protocol.
- **Digest.** User credentials are transmitted between clients and servers. A feature of HTTP 1.1, digest authentication does not use a password and therefore cannot be easily decrypted.
- **Integrated Windows Authentication (IWA).** With IWA, IIS uses standard Windows security to validate users. IWA is a Windows 2000-only authentication scheme that is only supported when using a Microsoft® Internet Explorer browser. IWA provides a way to transparently identify users and map them to their Web resources. NT Challenge/Response is a Windows NT-only authentication scheme.
- **Lightweight Directory Access Protocol (LDAP).** A standard for accessing directory information over the Internet, LDAP can be used to authenticate users. The benefit of using LDAP is that it retrieves information faster and requires a smaller client than a standard directory access protocol. You can also use a platform independent Application Programming Interface (API) to create applications that leverage the directory information.

Most Web servers allow more than one option to be used simultaneously. For example, you may want users to be able to roam freely at the higher levels of your reporting Web site. Unauthenticated users could then see what kind of information is available and click a link to apply for access to the application. Only when the user tries to access specific reports would the user be prompted for a logon ID and password.

For more detailed information about these authentication options, see your Web server documentation.

Web Server Environment Variables

Web servers populate variables that are available to programs executing on behalf of the browser client. There are a number of these variables, which store information such as the requestor's browser type, version, and IP address.

For example, the CGI variable `&REMOTE_USER` stores the value provided by the user during Web server authentication. This variable is populated only if Basic, Digest, or IWA authentication is being employed. A program can use this variable to adjust its behavior. Similarly, access to the program itself can be restricted from certain users based on the value of this parameter.

For more information about Web server variables, see Chapter 5, *WebFOCUS Administration*.

Web Server Access Control Using Aliases

Web servers can support a limited form of access control using an alias, a feature that maps a URL-based address to a physical directory location.

For example,

```
http://webfocus.anycorp.com/reports >> maps to >> g:\wwwroot\output
```

Web page aliases are generally read-only and may also be open to directory browsing. CGI aliases must allow execution and should not normally allow read access.

Web Server Confidentiality and Data Integrity

Once organizations have decided what degree of confidentiality and data integrity they want for their applications, two available options for consideration are:

- Directory Browsing
- Secure Sockets Layer (SSL)

Directory Browsing. As an added security measure, most Web servers have a directory browsing setting that, when enabled, allows users to view the contents of a Web server directory. When directory browsing is disabled (the default), the contents of this directory cannot be displayed. Although disabling directory browsing limits viewing, it does not prevent you from guessing the name of a file that is in the directory and then displaying it. For example, if you go to a Web site that does not allow directory browsing and correctly guess the URL of a page within that directory, then you would be given access to that Web page.

Secure Sockets Layer (SSL). One of the most common means to protect both the confidentiality and data integrity of Internet based transmissions is the Secure Sockets Layer (SSL) protocol. This is a feature based on the RSA encryption algorithm available in both 40-bit and 128-bit implementations. SSL is considered secure because the amount of time and CPU resources required for decryption makes the information safe for a long period of time. The algorithm is designed in such a way that if any data is altered it will fail its data integrity check upon receipt.

Database Security

Database security is generally maintained within the data subsystem. However, many databases offer security integration with the operating system security subsystems.

There are two major security schemes used by database systems:

- **Explicit Authorization** means that the database will authenticate the user's credentials and decide what type of access to grant. The user must provide a valid RDBMS user ID and password with each connection to the database.
- **Implicit Authorization** is when the Relational Database Management System (RDBMS) is configured to respect operating system security and is not required to authenticate the user. This type of access control is usually referred to as a trusted connection. The RDBMS typically manages access control based on the user's operating system logon ID, but this depends on the database vendor.

Note: Many RDBMS systems support both schemes, while others support only one. Refer to your RDBMS documentation for details.

CHAPTER 2

Providing Data Source Security: DBA

Topics:

- Introduction to Data Source Security
- Implementing Data Source Security
- Specifying an Access Type: The ACCESS Attribute
- Limiting Data Source Access: The RESTRICT Attribute
- Placing Security Information in a Central Master File
- Hiding Restriction Rules: The ENCRYPT Command
- WebFOCUS FOCEXEC Security

As Database Administrator, you can use WebFOCUS DBA security features to provide security for any WebFOCUS data source. You can use these security features to limit the number of records or reads a user can request in a report.

You can also use DBA security features to provide security for non-WebFOCUS data sources. However, the RESTRICT command (*Restricting an Existing FOCUS Data Source* on page 2-23) is not available for those data sources. Note that DBA security cannot protect a data source from non-WebFOCUS access.

Introduction to Data Source Security

The DBA facility provides a number of security options. You can:

- Limit the users who have access to a given data source using the USER attribute discussed in *Identifying Users With Access Rights: The USER Attribute* on page 2-6.
- Restrict a user's access rights to read, write, or update only using the ACCESS attribute discussed in *Specifying an Access Type: The ACCESS Attribute* on page 2-9.
- Restrict a user's access to certain fields or segments using the RESTRICT attribute discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 2-12.
- Ensure that only records that pass a validation test are retrieved using the RESTRICT attribute discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 2-12.
- Limit the values a user can write to the data source or you can limit which values a user can alter using the RESTRICT attribute discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 2-12.
- Point to passwords and restrictions stored in another Master File with the DBAFILE attribute discussed in *Placing Security Information in a Central Master File* on page 2-17.
- Use the WebFOCUS DBA exit routine to let an external security system set the WebFOCUS password. For more information, see Chapter 4, *WebFOCUS Custom Security Exits*.
- Place security on FOCEXECs, which is discussed in *WebFOCUS FOCEXEC Security* on page 2-24.

Implementing Data Source Security

You provide WebFOCUS security on a file-by-file basis. Implementing DBA security features is a straightforward process in which you specify:

- The names or passwords of WebFOCUS users granted access to a data source.
- The type of access the user is granted.
- The segments, fields, or ranges of data values to which the user's access is restricted.

The declarations (called security declarations) start following the END command in a Master File, and tell WebFOCUS that security is needed for the data source, and what type of security you want. Each security declaration can consist of one or several of the following attributes:

- The DBA attribute gives the name or password of the Database Administrator for the data source. The Database Administrator has unlimited access to the data source and its Master File.
- The USER attribute identifies a user as a legitimate user of the data source. Only users whose name or password is specified in the Master File of a WebFOCUS data source with security placed on it have access to that data source.
- The ACCESS attribute defines the type of access a given user has. The four types of access available are:
 - RW, which allows a user to both read and write to a data source.
 - R, which allows a user to read data in a data source only.
 - W, which allows a user to write new segment instances to a data source only.
 - U, which allows a user to update records in a data source only.
- The RESTRICT attribute specifies certain segments or fields to which the user is not granted access. It can also be used to restrict the data values a user can see or perform transactions on.
- The NAME and VALUE attributes are part of the RESTRICT declaration.

You describe your data source security by specifying values for these attributes in a comma-delimited format, just as you specify any other attribute in the Master File.

The word END on a line by itself in the Master File terminates the segment and field attributes and indicates that the access limits follow. If you place the word END in a Master File, it must be followed by at least a DBA attribute.

Example

Implementing Data Source Security in a Master File

The following is a Master File that uses security features:

```

FILENAME = PERS, SUFFIX = FOC,$
SEGMENT = IDSEG, SEGTYPE = S1,$
  FIELD = SSN           ,ALIAS = SSN       ,FORMAT = A9   ,,$
  FIELD = FULLNAME     ,ALIAS = FNAME    ,FORMAT = A40 ,,$
  FIELD = DIVISION     ,ALIAS = DIV      ,FORMAT = A8   ,,$
SEGMENT=COMPSEG, PARENT=IDSEG, SEGTYPE=S1,$
  FIELD = SALARY       ,ALIAS = SAL      ,FORMAT = D8   ,,$
  FIELD = DATE         ,ALIAS = DATE    ,FORMAT = YMD  ,,$
  FIELD = INCREASE     ,ALIAS = INC      ,FORMAT = D6   ,,$
END
DBA=JONES76,$
USER=TOM      ,ACCESS=RW, $
USER=BILL    ,ACCESS=R  ,RESTRICT=SEGMENT ,NAME=COMPSEG ,,$
USER=JOHN    ,ACCESS=R  ,RESTRICT=FIELD  ,NAME=SALARY  ,,$
              ,ACCESS=R  ,RESTRICT=FIELD  ,NAME=INCREASE,$
USER=LARRY   ,ACCESS=U  ,RESTRICT=FIELD  ,NAME=SALARY  ,,$
USER=TONY   ,ACCESS=R  ,RESTRICT=VALUE ,NAME=IDSEG  ,,$
              VALUE=DIVISION EQ 'WEST' ,,$
USER=MARY   ,ACCESS=W  ,RESTRICT=VALUE ,NAME=SALTEST,$
              VALUE=INCREASE+SALARY GE SALARY,$
              NAME=HISTTEST,
              VALUE=DIV NE ' ' AND DATE GT 0,$

```

Reference

Special Considerations for Data Source Security

When using the JOIN command, it is possible to bypass the DBA information in a data source. This is a security exposure created because in a JOIN structure the DBA information is read from the host Master File. This problem is solved by using the DBAFILE feature discussed in *Placing Security Information in a Central Master File* on page 2-17. All data sources in the joined structure will get security information as coded in the DBAFILE.

Identifying the DBA: The DBA Attribute

The first security attribute should be a password that identifies the Database Administrator. This password can be up to eight characters long. Since nothing else is needed, this line is terminated by the usual delimiter (,\$).

Note:

- Every data source having access limits must have a DBA.
- Groups of cross-referenced data sources must have the same DBA value.
- Partitioned data sources, which are read together in the USE command, must have the same DBA value.
- The Database Administrator has unlimited access to the data source and all cross-referenced data sources. Therefore, no field, segment, or value restrictions can be specified with the DBA attribute.
- You cannot encrypt and decrypt Master Files or restrict existing data sources without the DBA password.
- You should thoroughly test every security attribute before the data source is used. It is particularly important to test the VALUE limits to make sure they do not contain errors. Value tests are executed as if they were extra screening conditions or VALIDATE statements typed after each request statement. Since users are unaware of the value limits, errors caused by the value limits may confuse them.

Example

Identifying the DBA Using the DBA Attribute

```
DBA=JONES76,$
```

Procedure

Changing a DBA Password

The DBA has the freedom to change any of the security attributes. If you change the DBA password in the Master File, you must use the RESTRICT command for existing data sources (discussed in *Restricting an Existing FOCUS Data Source* on page 2-23) to inform each WebFOCUS data source affected by the change. Unless this is done, WebFOCUS will assume that the new description is an attempt to bypass the restriction rules. You use the following procedure for each data source affected:

1. Edit the Master File, changing the DBA value from old to new.

2. Issue the command:

```
SET PASS=old_DBA_value
```

3. Issue the command:

```
RESTRICT
filename
END
```

4. Issue the command:

```
SET PASS=new_value
```

Including the DBA Attribute in a HOLD File

With the SET HOLDSTAT command, you can identify a data source containing DBA information and comments to be automatically included in HOLD and PCHOLD Master Files. For more information about the SET HOLDSTAT command, see the *Developing Reporting Applications* manual.

Identifying Users With Access Rights: The USER Attribute

The USER attribute is a password that identifies the users who have legitimate access to the data source. A USER attribute cannot be specified alone; it must be followed by at least one ACCESS restriction (discussed in *Specifying an Access Type: The ACCESS Attribute* on page 2-9) to specify what sort of ACCESS the user is granted.

Before using a secured data source, a user must enter his or her password using the SET PASS command. If that password is not included in the Master File, the user is denied access to the data source. When the user does not have a password or has one that is inadequate for the type of access requested, the following message displays:

```
(FOC047) THE USER DOES NOT HAVE SUFFICIENT ACCESS RIGHTS TO THE FILE:  
filename
```

Syntax

How to Set the USER Attribute

Any user whose name or password is not declared in the Master File is denied access to that data source. The syntax of the USER attribute is

```
USER = name
```

where:

```
name
```

Is a password of up to eight characters for the user.

For example:

```
USER=TOM, . . .
```

You can specify a blank password. Such a password does not require the user to issue a SET PASS= command. A blank password may still have access limits and is convenient when a number of users have the same access rights. An example of setting a user's password to blank, and access to read only follows:

```
USER= , ACCESS=R,$
```

Establishing User Identity

A user must enter his or her password before using any WebFOCUS data source that has security specified for it. A single user may have different passwords in different files. For example, in file ONE the rights of password BILL apply, but in file TWO the rights of password LARRY apply. Use the SET PASS command to establish the passwords.

Syntax

How to Establish User Identity

```
SET {PASS|USER} = name [ [IN {file}* [NOCLEAR]} ] , name [IN file] ... ]
```

where:

name

Is the user's name or password.

file

Is the name of the Master File to which the password applies.

*

Indicates that *name* replaces all passwords active in all files.

NOCLEAR

Provides a way to replace all passwords in the list of active passwords while retaining the list.

Example Establishing User Identity

In the following example, the password TOM is in effect for all data sources that do not have a specific password designated for them:

```
SET PASS=TOM
```

For the next example, in file ONE the password is BILL, and in file TWO the password is LARRY. No other files have passwords set for them:

```
SET PASS=BILL IN ONE, LARRY IN TWO
```

Here, all files have password SALLY, except files SIX and SEVEN, which have password DAVE:

```
SET PASS=SALLY, DAVE IN SIX  
SET PASS=DAVE IN SEVEN
```

The password is MARY in file FIVE, and FRANK in all other files:

```
SET PASS=MARY IN FIVE, FRANK
```

A list of the files for which a user has set specific passwords is maintained. To see the list of files, issue:

```
? PASS
```

When the user sets a password IN * (all files), the list of active passwords collapses to one entry with no associated file name. To retain the file name list, use the NOCLEAR option.

In the next example, the password KEN replaces all passwords active in all files, and the table of active passwords is folded to one entry:

```
SET PASS=KEN IN *
```

In the following, MARY replaces all passwords in the existing table of active passwords (which consists of files NINE and TEN) but FRANK is the password for all other files. The option NOCLEAR provides a shorthand way to replace all passwords in a specific list:

```
SET PASS=BILL IN NINE, TOM IN TEN  
SET PASS=MARY IN * NOCLEAR, FRANK
```

Note: The FIND function does not work with COMBINED data sources secured with different passwords.

Users must issue their passwords using the SET PASS command during each session in which they use a secured data source. They may issue their passwords at any time before using the data source and can issue a different password afterward to access another data source.

Specifying an Access Type: The ACCESS Attribute

The ACCESS attribute specifies what sort of access a user is granted. Every security declaration, except the DBA declaration, must have a USER attribute and an ACCESS attribute.

The following is a complete security declaration, consisting of a USER attribute and an ACCESS attribute.

```
USER=TOM, ACCESS=RW, $
```

This declaration gives Tom read and write (for adding new segment instances) access to the data source.

You can assign the ACCESS attribute one of four values. These are:

ACCESS=R	Read only
ACCESS=W	Write only
ACCESS=RW	Read the data source and write new segment instances
ACCESS=U	Update only

Access levels affect what kind of commands a user can issue. Before you decide what access levels to assign to a user, you must consider what commands that user will need. If a user does not have sufficient access rights to use a given command, the following message displays:

```
(FOC047) THE USER DOES NOT HAVE SUFFICIENT ACCESS RIGHTS TO THE FILE:  
filename
```

ACCESS levels determine what a user can do to the data source. You use the RESTRICT attribute (discussed in *Limiting Data Source Access: The RESTRICT Attribute* on page 2-12) to limit the fields, values, or segments to which a user has access. Every USER attribute must be assigned an ACCESS attribute. The RESTRICT attribute is optional; without it, the user has unlimited access to fields and segments within the data source.

Types of Access

The type of access granting use of various WebFOCUS commands is shown in the following table. When more than one type of access is shown, any type of access marked will allow the user at least some use of that command. Often, however, the user will be able to use the command in different ways, depending on the type of access granted.

Command	R	W	RW	U	DBA
CHECK	X	X	X	X	X
CREATE			X		X
DECRYPT					X
DEFINE	X		X		X
ENCRYPT					X
MATCH	X		X		X
REBUILD			X		X
RESTRICT					X
TABLE	X		X		X

CHECK Command. Users without the DBA password or read/write access are allowed limited access to the CHECK command. However, when the HOLD option is specified, the warning ACCESS LIMITED BY PASSWORD is produced, and restricted fields are propagated to the HOLD file depending on the DBA RESTRICT attribute. Refer to *Limiting Data Source Access: The RESTRICT Attribute* on page 2-12 for more information on the RESTRICT attribute.

CREATE Command. Only users with the DBA password or read/write (RW) access rights can issue a CREATE command.

DECRYPT Command. Only users with the DBA password can issue a DECRYPT command.

DEFINE Command. As with all reporting commands, a user need only have an access of R (read only) to use the DEFINE command. An access of R permits the user to read records from the data source and prepare reports from them. The only users who cannot use the DEFINE command are those whose access is W (write only) or U (update only).

ENCRYPT Command. Only users with the DBA password can use the ENCRYPT command.

REBUILD Command. Only users with the DBA password or read/write (RW) access rights can issue the REBUILD command. This command is only for FOCUS data sources.

RESTRICT Command. Only users with the DBA password may use the RESTRICT command.

TABLE or MATCH Command. A user who has access of R or RW may use the TABLE command. Users with access of W or U may not.

Reference

RESTRICT Attribute Keywords

The RESTRICT attribute keywords affect the resulting HOLD file as follows:

FIELD

Fields named with the NAME parameter are not included in the HOLD file.

SEGMENT

The segments named with the NAME parameter are included, but fields in those segments are not.

SAME

The behavior is the same as for the user named in the NAME parameter.

NOPRINT

Fields named in the NAME or SEGNAME parameter are included since the user can reference these.

VALUE

Fields named in the VALUE parameter are included since the user can reference these.

If you issue the CHECK command with the PICTURE option, the RESTRICT attribute keywords affect the resulting picture as follows:

FIELD

Fields named with the NAME parameter are not included in the picture.

SEGMENT

The boxes appear for segments named with the NAME parameter, but fields in those segments do not.

SAME

The behavior is the same as for the user named in the NAME parameter.

NOPRINT

This option has no effect on the picture.

VALUE

This option has no effect on the picture.

Limiting Data Source Access: The RESTRICT Attribute

The ACCESS attribute determines what a user can do with a data source. The optional RESTRICT attribute further restricts a user's access to certain fields, values, or segments.

Syntax

How to Limit Data Source Access

```
...RESTRICT=level, NAME={name|SYSTEM} [, VALUE=test], $
```

where:

level

Can be one of the following:

FIELD specifies that the user cannot access the fields named with the NAME parameter.

SEGMENT specifies that the user cannot access the segments named with the NAME parameter.

SAME specifies that the user has the same restrictions as the user named in the NAME parameter. No more than four nested SAME users are valid.

NOPRINT specifies that the field named in the NAME or SEGMENT parameter can be mentioned in a request statement, but will not be displayed.

name

Is the name of the field or segment you wish to restrict. When used after NOPRINT, this can be only a field name. NAME=SYSTEM, which can only be used only with value tests, restricts every segment in the data source, including descendant segments. Multiple fields or segments can be specified by issuing the RESTRICT attribute several times for one user.

VALUE

Specifies that the user can have access to only those values that meet the test described in the *test* parameter.

test

Is the value test that the data must meet before the user can have access to it.

Example

Limiting Data Source Access

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG, $
```

Restricting Access to a Field or a Segment

The RESTRICT attribute identifies the segments or fields that the user will not be able to access. Anything not named in the RESTRICT attribute will be accessible.

Without the RESTRICT attribute, the user has access to the entire data source. Users may be limited to reading, writing, or updating new records, but every record in the data source is available for the operation.

Syntax

How to Restrict Access to a Field or a Segment

```
... RESTRICT=level, NAME=name, $
```

where:

level

Can be one of the following:

FIELD specifies that the user cannot access the fields named with the NAME parameter.

SEGMENT specifies that the user cannot access the segments named with the NAME parameter.

SAME specifies that the user has the same restrictions as the user named in the NAME parameter.

NOPRINT specifies that the field named in the NAME or SEGMENT parameter can be mentioned in a request statement but will not be displayed. When used after NOPRINT, NAME can be only a field name.

name

Is the name of the field or segment you wish to restrict. When used after NOPRINT, this can be only a field name.

NAME=SYSTEM, which can be used only with value tests, restricts every segment in the data source, including descendant segments. Multiple fields or segments can be specified by issuing the RESTRICT attribute several times for one user.

Note:

- If a field or segment is mentioned in the NAME attribute, it cannot be retrieved by the user. If such a field or segment is mentioned in a request statement, it will be rejected as beyond the user's access rights. With NOPRINT, the field or segment can be mentioned, but the data will not be displayed. The data will appear as blanks for alphanumeric format or zeroes for numeric fields.
- You can restrict multiple fields or segments by providing multiple RESTRICT statements. For example, if you wish to restrict Harry from using both field A and segment B, you issue the following access limits:

```
USER=HARRY, ACCESS=R, RESTRICT=FIELD , NAME=A, $  
RESTRICT=SEGMENT, NAME=B, $
```

- You can restrict as many segments and fields as you like.
- Using RESTRICT=SAME is a convenient way to reuse a common set of restrictions for more than one password. If you specify RESTRICT=SAME and provide a user name or password as it is specified in the USER attribute for the NAME value, the new user will be subject to the same restrictions as the one named in the NAME attribute. You can then add additional restrictions, as they are needed.

Example

Restricting Access to a Segment

In the following example, Bill has read-only access to everything in the data source except the COMPSEG segment:

```
USER=BILL ,ACCESS=R ,RESTRICT=SEGMENT ,NAME=COMPSEG, $
```

Example

Reusing a Common Set of Access Restrictions

In the following example, both Sally and Harry have the same access privileges as BILL. In addition, Sally is not allowed to read the SALARY field.

```
USER=BILL ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'WEST', $  
USER=SALLY ,ACCESS=R ,RESTRICT=SAME ,NAME=BILL,  
RESTRICT=FIELD ,NAME=SALARY, $  
USER=HARRY ,ACCESS=R ,RESTRICT=SAME ,NAME=BILL, $
```

Note: A restriction on a segment also affects access to its descendants.

Restricting Access to a Value

You can also restrict the values to which a user has access by providing a test condition in your RESTRICT attribute. The user is restricted to using only those values that satisfy the test condition.

You can restrict values in one of two ways: you can restrict the values the user can read from the data source, or you can restrict what the user can write to a data source. These restrictions are two separate functions: one does not imply the other. You use the ACCESS attribute to specify whether the values the user reads or the values the user writes are restricted.

You restrict the values a user can read by setting ACCESS=R and RESTRICT=VALUE. This type of restriction prevents the user from seeing any data values other than those that meet the test condition provided in the RESTRICT attribute. A RESTRICT attribute with ACCESS=R functions as an involuntary IF statement in a report request. Therefore, the syntax for ACCESS=R value restrictions must follow the rules for an IF test in a report request.

Syntax

How to Restrict Values a User Can Read

```
...ACCESS=R, RESTRICT=VALUE, NAME=name, VALUE=test,$
```

where:

name

Is the name of the segment on which you are performing the tests. To specify all segments in the data source, specify NAME=SYSTEM.

test

Is the test being performed.

Example

Restricting Values a User Can Read

```
USER=TONY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'WEST' , $
```

With this restriction, Tony can see records from only the western division.

You type the test expression after VALUE=. The syntax of the test condition is the same as that used by the TABLE command to screen records, except the word IF does not precede the phrase. (Screening conditions in the TABLE command are discussed in the *Creating Reports With WebFOCUS Language* manual.) Should several fields have tests performed on them, separate VALUE attributes must be provided. Each test must name the segment to which it applies. For example:

```
USER=DICK ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'EAST' OR 'WEST' , $  
NAME=IDSEG,  
VALUE=SALARY LE 10000 , $
```

If a single test condition exceeds the allowed length of a line, it can be provided in sections. Each section must start with the attribute VALUE= and end with the terminator (,\$). For example:

```
USER=SAM, ACCESS=R, RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'EAST' OR 'WEST' , $  
VALUE=OR 'NORTH' OR 'SOUTH' , $
```

Note: The second and subsequent lines of a value restriction must begin with the keyword OR.

You can apply the test conditions to the parent segments of the data segments on which the tests are applicable. Consider the following example:

```
USER=DICK ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'EAST' OR 'WEST' , $  
NAME=IDSEG,  
VALUE=SALARY LE 10000 , $
```

The field named SALARY is actually part of a segment named COMPSEG. Since the test is specified with NAME=IDSEG, however, the test is made effective for requests on its parent, IDSEG. In this case, the request PRINT FULLNAME would print the full names of only people who meet this test, that is, whose salary is less than or equal to \$10,000, even though the test is performed on a field that is part of a descendant segment of IDSEG. If, however, the test was made effective on COMPSEG, that is, NAME=COMPSEG, then the full name of everyone in the data source could be retrieved, but with the salary information of only those meeting the test condition.

Example

Restricting Both Read and Write Values for a User

```
USER=TILLY ,ACCESS=R ,RESTRICT=VALUE ,NAME=IDSEG,  
VALUE=DIVISION EQ 'NORTH' , $  
ACCESS=W ,RESTRICT=VALUE ,NAME=DIVTEST,  
VALUE=DIVISION EQ 'NORTH' , $
```

Placing Security Information in a Central Master File

The DBAFILE attribute enables you to place all of the passwords and restrictions for many Master Files in one central file. Each individual Master File points to this central control file. Groups of Master Files with the same DBA password may share a common DBAFILE which itself has the same DBA password.

There are several benefits to this technique. The primary ones are:

- Passwords have to be stored only once when they are applicable to a group of data sources. This simplifies password administration.
- Data sources with different DBA passwords can now be JOINed or COMBINED. In addition, individual DBA information remains in effect for each data source in a JOIN or COMBINE.

The central DBAFILE is a standard Master File. Other Master Files can use the password and security restrictions listed in the central file by specifying its file name with the DBAFILE attribute.

Note:

- All Master Files that specify the same DBAFILE have the same DBA password.
- The central DBAFILE may include additional attributes before the END statement that signifies the presence of DBA information. The DBA password in the DBAFILE is the same as the password in all the Master Files that refer to it. This prevents individuals from substituting their own security. All of these Master Files should be encrypted.
- The DBAFILE may contain a list of passwords and restrictions following the DBA password. These passwords apply to all data sources that reference this DBAFILE. In the example above, PASS=BILL, with ACCESS=R (read only), applies to all data sources that contain the attribute DBAFILE=FOUR.
- After the common passwords, the DBAFILE may specify data source-specific passwords and additions to general passwords. You implement this feature by including FILENAME attributes in the DBA section of the DBAFILE (for example, FILENAME=TWO). Consult *File Naming Requirements for DBAFILE* on page 2-19 for additional information about the FILENAME attribute.
- Data source-specific restrictions override general restrictions for the specified data source. In the case of a conflict, passwords in the FILENAME section take precedence. For example, a DBAFILE might contain ACCESS=RW in the common section, but specify ACCESS=R for the same password by including a FILENAME section for a particular data source.
- Value restrictions accumulate; all value restrictions must be satisfied before retrieval. In the preceding example, note the two occurrences of PASS=JOE. JOE is a common password for all data sources, but in FILENAME=THREE it carries an extra restriction, RESTRICT=..., which applies only to data source THREE.

Syntax

How to Place Security Attributes in a Central Master File

```
END
DBA=dbaname, DBAFILE=filename , $
```

where:

dbaname

Is the same as the dbaname in the central file.

filename

Is the name of the central file.

You can specify passwords and restrictions in a DBAFILE that apply to every Master File that points to that DBAFILE; you can also include passwords and restrictions for specific Master Files by including FILENAME attributes in the DBAFILE.

Example

Placing Security Attributes in a Central Master File

The following example shows a group of Master Files that share a common DBAFILE named FOUR:

```
ONE MASTER
FILENAME=ONE
.
.
END
DBA=ABC, DBAFILE=FOUR, $

TWO MASTER
FILENAME=TWO
.
.
END
DBA=ABC, DBAFILE=FOUR, $

THREE MASTER
FILENAME=THREE
.
.
END
DBA=ABC,
DBAFILE=FOUR, $

FOUR MASTER
FILENAME=FOUR, $
SEGNAME=mmmmm, $
FIELDNAME=fffff, $
END
DBA=ABC, $
    PASS=BILL, ACCESS=R, $
    PASS=JOE, ACCESS=R, $
FILENAME=TWO, $
    PASS=HARRY, ACCESS=RW, $
FILENAME=THREE, $
    PASS=JOE, ACCESS=R, RESTRICT=... , $
    PASS=TOM, ACCESS=R, $
```

File Naming Requirements for DBAFILE

When a DBAFILE includes a FILENAME attribute for a specific Master File, the FILENAME attribute in the referencing Master File must be the same as the FILENAME attribute in the DBA section of the DBAFILE. This prevents users from renaming a Master File to a name not known by the DBAFILE.

Example

DBAFILE Naming Conventions

```
ONE MASTER
FILENAME=XONE
.
.
.
END
DBA=ABC, DBAFILE=FOUR, $
```

```
FOUR MASTER
FILENAME=FOUR
.
.
.
END
DBA=ABC, $
.
.
.
FILENAME=XONE, $
.
.
.
```

ONE MASTER is referred to in requests as TABLE FILE ONE. However, both ONE MASTER and the DBA section of the DBAFILE, FOUR MASTER, specify FILENAME=XONE.

Connection to an Existing DBA System With DBAFILE

If there is no mention of the new attribute, DBAFILE, there will be no change in the characteristics of an existing system. In the current system, when a series of data sources is JOINed, the first data source in the list is the controlling data source. Its passwords are the only ones examined. For a COMBINE, only the last data source's passwords take effect. All data sources must have the same DBA password.

In the new system, the DBA sections of all data sources in a JOIN or COMBINE are examined. If DBAFILE is included in a Master File, then its passwords and restrictions are read. To make the DBA section of a data source active in a JOIN list or COMBINE, specify DBAFILE for that data source.

Once you start to use the new system, you should convert all of your Master Files. For Database Administrators who want to convert existing systems but do not want a separate physical DBAFILE, the DBAFILE attribute can specify the data source itself.

Example Connecting to an Existing DBA System With DBAFILE

```
FILENAME=SEVEN,  
  SEGNAME=..  
  FIELDNAME=...  
  .  
  .  
  .  
END  
DBA=ABC,DBAFILE=SEVEN,$      (OR DBAFILE= , $)  
  PASS=...  
  PASS=...
```

Combining Applications With DBAFILE

Since each data source now contributes its own restrictions, you can now JOIN and COMBINE data sources that come from different applications and have different DBA passwords. The only requirement is a valid password for each data source. You can therefore grant access rights for one application to an application under the control of a different DBA by assigning a password in your system.

You can assign screening conditions to a data source that are automatically applied to any report request that accesses the data source. See the *Creating Reports With WebFOCUS Language* manual for details.

Summary of Security Attributes

The following is a list of all the security attributes used in WebFOCUS:

Attribute	Alias	Maximum Length	Meaning
DBA	DBA	8	Value assigned is code name of the Database Administrator (DBA) who has unrestricted access to the data source.
USER	PASS	8	Values are arbitrary code names, identifying users for whom security restrictions will be in force.
ACCESS	ACCESS	8	Levels of access for this user. Values are: R read only W write new segments only RW read and write U update values only
RESTRICT	RESTRICT	8	Types of restrictions to be imposed for this access level. Values are: SEGMENT FIELD VALUE SAME NOPRINT
NAME	NAME	66	Name of segment or field restricted or of the program to be called.
VALUE	VALUE	80	Test expression which must be true when RESTRICT=VALUE is the type of limit.
DBAFILE	DBAFILE	8	Names the Master File that contains passwords and restrictions to use.

Hiding Restriction Rules: The ENCRYPT Command

Since the restriction information for a WebFOCUS data source is stored in its Master File, you will want to encrypt the Master File in order to prevent users from examining the restriction rules. Only the Database Administrator can encrypt a description. Thus, you must set PASS=DBAname before you issue the ENCRYPT command. The syntax of the ENCRYPT command varies from operating system to operating system.

Syntax

How to Hide Restriction Rules: ENCRYPT Command

```
ENCRYPT FILE filename
```

where:

```
filename
```

Is the name of the file to be encrypted.

Example

Encrypting and Decrypting a Master File

The following is an example of the complete procedure:

```
SET PASS=JONES76  
ENCRYPT FILE PERS
```

The process can be reversed if you wish to change the restrictions. The command to restore the description to a readable form is DECRYPT.

The DBA password must be issued with the SET command before the file can be decrypted. For example:

```
SET PASS=JONES76  
DECRYPT FILE PERS
```

Encrypting Data

You may also use the ENCRYPT command within the Master File to encrypt some or all of its segments. When encrypted files are stored on their external media (disk or tape) they are secure from unauthorized examination.

Encryption takes place on the segment level; that is, the entire segment is encrypted. The request for encryption is made in the Master File by setting the attribute ENCRYPT to ON.

Example

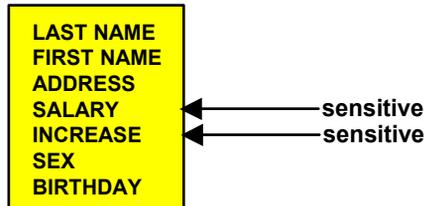
Encrypting Data

```
SEGMENT=COMPSEG, PARENT=IDSEG, SEGTYPE=S1, ENCRYPT=ON,$
```

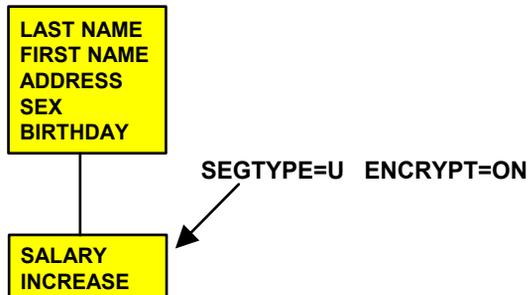
You must specify the ENCRYPT attribute before you enter any data in the data source. The message NEW FILE... must appear when the encryption is first requested. Encryption cannot be requested later by a change to the Master File and cannot be removed once it has been requested and any data has been entered in the data source.

Performance Considerations for Encrypted Data

There is a small loss in processing efficiency when data is encrypted. You can minimize this loss by grouping the sensitive data fields together on a segment and making them a separate segment of SEGTYPE=U, unique segment, beneath their original segment. For example, suppose the data items on a segment are:



They should be grouped as:



Restricting an Existing FOCUS Data Source

If you have existing files to which you want to add security limitations, you need to use the RESTRICT command.

(**Note:** This is not the RESTRICT attribute described in *Limiting Data Source Access: The RESTRICT Attribute* on page 2-12.)

If an existing WebFOCUS data source has no DBA rules and you want to add DBA rules, perform the following:

1. Edit the Master File and add the DBA rules.
2. Issue the RESTRICT command to write the DBA password to the data source.

Note: The RESTRICT command cannot be used for non-WebFOCUS data sources.

Setting a Password Externally

Passwords can also be set automatically by an external security system such as RACF[®], CA-ACF2[®], or CA-Top Secret[®]. Passwords issued this way are set when WebFOCUS is first entered and may be permanent (that is, not alterable by subsequent SET USER, SET PASS or -PASS commands); or they may be default passwords that can be subsequently overridden; or they may be permanent for some users, defaults for other users, and not set at all for yet other users.

The advantage of setting WebFOCUS passwords externally is that the password need not be known by the user, does not require prompting, and does not have to be embedded in a PROFILE FOCEXEC or an encrypted FOCEXEC.

Passwords set this way must match the passwords specified in the Master Files of the data sources being accessed.

WebFOCUS FOCEXEC Security

Most data security issues are best handled by WebFOCUS DBA exit routines. For more information about WebFOCUS DBA exit routines, see Chapter 4, *WebFOCUS Custom Security Exits*. However, an additional data security facility that can be incorporated within Dialogue Manager is to encrypt and decrypt FOCEXECs.

Encrypting and Decrypting a FOCEXEC

You may want to keep the actual text of a stored FOCEXEC confidential while allowing users to execute the FOCEXEC. You may want to do this either because there is confidential information stored in the FOCEXEC or because you do not want the FOCEXEC changed by unauthorized users. You can protect a stored FOCEXEC from unauthorized users with the ENCRYPT command.

Any user can execute an encrypted FOCEXEC, but you must decrypt the FOCEXEC to view it. Only a user with the DBA password can decrypt the FOCEXEC.

Syntax

How to Encrypt and Decrypt a FOCEXEC

You use the following procedure to encrypt the FOCEXEC named SALERPT:

```
SET PASS = DOHIDE  
ENCRYPT FILE SALERPT FOCEXEC
```

You use the following procedure to decrypt the FOCEXEC named SALERPT:

```
SET PASS = DOHIDE  
DECRYPT FILE SALERPT FOCEXEC
```

CHAPTER 3

WebFOCUS Security

Topics:

- WebFOCUS Reporting Server Security
- WebFOCUS Client Security
- Managed Reporting Environment (MRE) Security
- ReportCaster Security

WebFOCUS security concerns the following components:

- **WebFOCUS Reporting Server.** Processes a report request, retrieves and formats data, and returns report output.
- **WebFOCUS Client.** Web server-based component (CGI, ISAPI, or servlet).

WebFOCUS may optionally include the following components:

- **Managed Reporting Environment.** Enables administrators to define multiple reporting domains and control user access to data. Installed with the WebFOCUS Client.
- **ReportCaster.** Facilitates scheduled report distribution.

One of the strengths of WebFOCUS is its ability to operate in a high security environment. A properly configured and managed implementation of WebFOCUS provides an acceptable level of security for any organization. The following sections describe the security concepts for each WebFOCUS component.

WebFOCUS Reporting Server Security

There are three important security concepts that affect WebFOCUS Reporting Server behavior:

- Security settings
- Deployment options
- Processing modes

The following sections describe these concepts.

WebFOCUS Reporting Server Security Settings

WebFOCUS Reporting Server security runs in one of two modes:

- **Security On** means that the user is authenticated by a valid user ID and password, and that the agent that gets started has the security profile of the authenticated user.
- **Security Off** means that there is no authentication and that all agents created by the server will run with the security profile of the user that started the server.

Important:

When security is off, other security measures (such as Web server security and firewalls) should be in place in order to ensure that any unwanted access is denied.

For details on how to issue this setting, see the iWay Server documentation for your platform.

The security behavior of the user's agent process varies depending on whether WebFOCUS Reporting Server security is on or off. Make sure you match the following settings with your requirements.

Security Setting	Effective User of Workspace Manager Process	Effective User of Individual Agent Processes
ON	<p>An authorized user ID on OS/390.</p> <p>Root user on UNIX (accomplished using the Set User Privilege).</p> <p>Selectable user ID with SYSPRIV on OpenVMS.</p> <p>Typically the system account on Windows NT®.</p>	<p>Private deployment. Operating system user ID of the authenticated user. Limits access to the privileges of that user's security profile.</p> <p>Pooled deployment. Configurable user ID.</p> <p>For more information, see <i>WebFOCUS Reporting Server Deployment Options</i> on page 3-4.</p>
OFF	<p>Operating system user ID of the user who started Workspace Manager.</p> <p>The assigned user ID should only have privileges to access resources associated with the WebFOCUS Reporting Server.</p> <p>Important:</p> <p>While this is a flexible feature, please keep in mind that it could have unseen ramifications. On Windows NT, for example, if you want security off, the WebFOCUS Reporting Server should never be started with an Administrator or system account, since all connecting users would by default have system privileges. For more information, see the iWay installation and configuration documentation for your specific platform.</p>	<p>The same as the Effective User of Workspace Manager Process. All users have the same access rights.</p>

Some administrators believe that running the WebFOCUS Reporting Server from a privileged account such as the "root" (or system) account is a security exposure. If you do not run your server with a privileged account you can enable security on the WebFOCUS Reporting Server by using security exits (for details, see Chapter 4, *WebFOCUS Custom Security Exits*). However, you cannot use the built-in SECURITY=ON setting because the WebFOCUS Reporting Server needs "privileged" authority to access the APIs that authenticate you to the operating system. For more detailed information, see the *WebFOCUS and ReportCaster Installation and Configuration* manual for your platform.

WebFOCUS Reporting Server Deployment Options

There are two ways to deploy WebFOCUS Reporting Server resources:

- **Private Deployment.** Indicates that for each client connection request, a dedicated application agent is assigned. The privileges of each application agent depend on the security setting of the WebFOCUS Reporting Server.

When security is on, authentication is processed for every client logging on to the server. If the client user ID passes Windows NT/2000 authentication, the agent will have the Windows NT/2000 privileges associated with that user ID. This is the default.

When security is off, there is no user identification and authentication required. Requests are processed as the WebFOCUS Reporting Server user ID.

- **Pooled Deployment.** Allows a predetermined number of application agents to support a large community of users, provided that the application is designed to support the small Logical Unit of Work (LUW) concept. These application agents establish their application environments on startup and maintain their environments, including database connections between LUWs, for user connections.

Pooled deployment executes the global server profile only. Therefore, each application agent inherits the privileges of one user account. Determining the user account ID that all application agents will share depends on what operating system you are using and whether or not you have set external security on or off. Each user in the pool shares the user ID of the pooled ID, which was supplied during configuration.

WebFOCUS Reporting Server Processing Modes

For WebFOCUS Reporting Servers running with security on, there are two possible modes of operation:

- **Explicit Verification Processing** is when a user ID and password are required to connect to the server. This is the default setting.
- **Already Verified Processing (AVP)** requires that the WebFOCUS Client send only the user ID, along with a special flag indicating that authentication has already taken place.

AVP is commonly used with Web server authentication (for more information, see *Web Server Authentication* in Chapter 1, *Security Overview*), where the value of &REMOTE_USER is set for IBIC_user, or with a WebFOCUS callable exit program, where the value of IBIC_user is returned by the program.

With the following setting in the `ibidir.wfs` file, an already authenticated Web server user ID is used for the connection to the WebFOCUS Reporting Server, and no password is required.

```
IBIC_user=&REMOTE_USER  
IBIC_pass="TRUSTED"
```

Note:

- AVP mode is not supported inbound to a WebFOCUS Reporting Server on Windows NT/2000. However, AVP mode is supported between NT WebFOCUS Client and any other supported WebFOCUS Reporting Server platform.
- If you are using Microsoft Internet Explorer and Internet Information Server (IIS) with the Integrated Windows Authentication (IWA) authentication scheme, the `&REMOTE_USER` value may include the client's security domain (DOMAIN/user ID). Use `&WF_REMOTE_USER` instead of `&REMOTE_USER` to always obtain only the user ID.

Pre-Started Agents

Pre-started agents are an optional start-up setting of the WebFOCUS Reporting Server that allows for the quicker initialization of WebFOCUS Reporting Server agents. Both private and pooled deployment schemes support pre-started agents. Pre-started pooled agents are started with the user ID of the "pooled user" and maintain that context upon connection.

Regardless of whether WebFOCUS Reporting Server Security is on or off, pre-started private agents are started under the same user ID as the WebFOCUS Reporting Server. When server security is on, that is usually a "system-like" ID. However, as soon as a user is authenticated and connected with a pre-started agent, that agent's user context "impersonates" that of the connecting user, thus limiting the agent to that user's access privileges.

Windows NT/2000 Domain Security Considerations

On Windows NT and Windows® 2000, a Primary Domain Controller (PDC) server is typically used to manage the user account information for a given domain (for example, \main-domain). Users then typically identify themselves to servers in that domain with only their user ID since the domain name is assumed.

When users want to identify themselves to a server that is not in their domain (for example, \resource-domain), they must qualify their user ID (for example, \main-domain\userid). A trust relationship must also exist between the domains to support this type of authentication.

WebFOCUS only supports NT domain security if the following conditions are true:

- An NT trust relationship has been properly defined between the two domains.
- The remote user ID (or an NT group that the user is in) has been granted the logon locally NT user rights to the server hosting the WebFOCUS Reporting Server.

WebFOCUS Reporting Server Administrator ID

During installation, you will be prompted for the WebFOCUS Reporting Server Administrator user ID. This is assumed to be the currently logged on user, but you may change it on the installation screen. The value provided is written to the `server_admin_id` keyword in the `srvrel\wfs\bin\edaserve.cfg` file (where *rel* is the WebFOCUS release). When you log on to the server Web console with this user ID, you will be able to perform administrator tasks, such as restarting the WebFOCUS Reporting Server.

WebFOCUS Client Security

The WebFOCUS Client may be implemented as a Common Gateway Interface (CGI) program, an Internet Server API (ISAPI) program (for Microsoft® IIS Web servers only), or as a Java™ servlet. The WebFOCUS Client can be customized with user-written security exits, which can integrate WebFOCUS with a Lightweight Directory Access Protocol (LDAP) server. For more information, see Chapter 4, *WebFOCUS Custom Security Exits*.

Connecting to a Secure WebFOCUS Reporting Server

For the WebFOCUS Client to connect to a secure WebFOCUS Reporting Server, it needs to provide a valid user ID and password using the variables `IBIC_user` and `IBIC_pass`.

For example,

```
http://wserver1/cgi-bin/ibiweb.exe?IBIF_ex=TESTPRG&IBIC_user=USER1&IBIC_pass=PASS1
```

While this type of parameter passing is possible it is typically not practical, since it needs to be passed on every request and it is exposed as part of the query string.

Setting WebFOCUS Security Defaults

WebFOCUS maintains a number of default settings, including default security settings (see your *WebFOCUS and ReportCaster Installation and Configuration* manual for details). The `.wfs` files included with the WebFOCUS Client are control files that define the behavior of the client.

Note: You can ensure that users are authenticated by means of a signon page. For an example, see *Allowing for the Conditional Checking of Variables* in Chapter 5, *WebFOCUS Administration*.

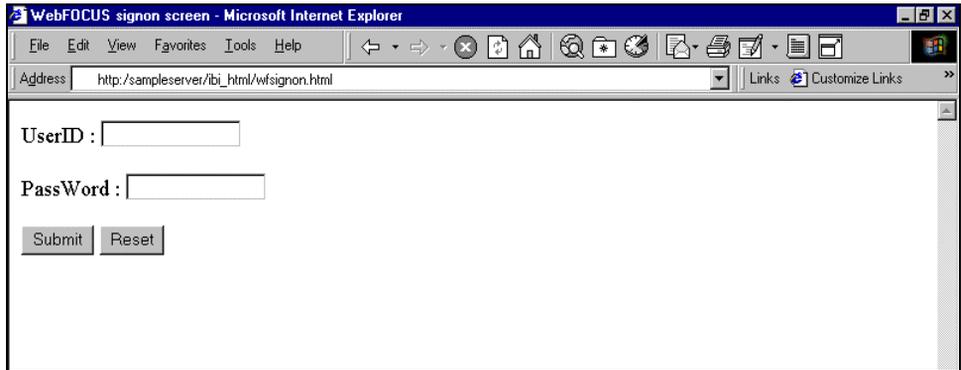
When a user connects to the WebFOCUS Reporting Server, if a value for any of the required fields is not supplied by the incoming request, the WebFOCUS Client will look in the `.wfs` file for the defaults.

If you want WebFOCUS Reporting Server security on without users having to supply individual credentials, you can substitute a valid user ID (`IBIC_user`) and password (`IBIC_pass`) in the `ibidir.wfs` file. You can also set `IBI_REPORT_USER` and `IBI_REPORT_PASS`, but these settings can be overridden by `IBIC_user` and `IBIC_pass`.

The security credentials of this default user ID can then be set appropriately to control access to the application. All users that do not supply these variables will use the defaults.

WebFOCUS Signon Processing

The WebFOCUS Client allows a user to set security credentials once, after which all subsequent requests from the WebFOCUS Client to the WebFOCUS Reporting Server will automatically be supplied with these credentials. To do this, the user ID and password are usually provided by the user on a Web page that uses an HTML <form> tag. When the form is submitted with the hidden variable IBIWF_action set to "WF_SIGNON," the WebFOCUS Client encrypts the signon information and, using its built-in iWay connector, transmits the information to a WebFOCUS Reporting Server for authentication. The following sample logon form displays during WebFOCUS installation and can be customized to meet the needs of each customer's application.



```
<form action="cgi-bin/ibi_cgi/webapi.dll" method="POST">
user ID :
<input type="text" name="IBIC_user" value="">
PassWord :
<input type="password" name="IBIC_pass" value="">
<input type="submit" name="submit" value="Submit">
<input type="reset" name="rest" value="Reset">
<input type="hidden" name="IBIWF_action" value="WF_SIGNON">
</form>
```

If the logon information is authenticated successfully by the WebFOCUS Reporting Server, the WebFOCUS Client stores the encrypted information in a browser cookie and redirects the user to the URL specified by the WF_SIGNON_MESSAGE variable. If this variable is not specified on the logon form, the value found in ibidir.wfs is used. While WebFOCUS provides a sample logon page (wfsignon.html), you can enhance and modify this page to suit your applications. You can also bypass this page completely. For example, you can use a WebFOCUS callable exit or set the user ID and password in a .wfs file.

WebFOCUS Cookies

Web servers support HTTP variables, which are passed between the Web server and Web browser in the HTTP request/response header. These variables typically define the environment of the Web browser. For example, HTTP_USER_AGENT defines which browser manufacturer and version of the browser is being run. Another HTTP variable is HTTP_COOKIE, which differs from other variables because it remains persistent for each request. The WF_SIGNON action updates this variable to include an array of information (the “cookie”), including WF_COOKIE. Within WF_COOKIE, the valid parameters to be set are IBIC_user, IBIC_pass, IBIC_server, IBID_user, IBID_pass, and IBID_server. For more information, see the *WebFOCUS Developing Reporting Applications* manual.

However, since this is sensitive information, there are certain restrictions. For example, cookies that are set are written to a cookie file on the user’s computer. This happens when they are set with an expiration that is sometime in the future. The security cookie set by WebFOCUS is encrypted. It always has an expiration of end-of-session and expires when the browser closes. It is not stored on the user’s computer. For more information about utilizing an alternate encryption routine (MRCEXT3 exit), see Chapter 4, *WebFOCUS Custom Security Exits*.

If users want to change or override an existing set of credentials, all they have to do is resubmit the logon page. This will override the current values with the values specified in the new request.

Reference **WebFOCUS Cookie Variables**

Use the following variables in site.wfs to control the behavior of a WebFOCUS cookie:

IBI_COOKIE_DOMAIN. The value of the domain field of a cookie. This value will restrict where the cookie is sent. For example, if you specify a particular domain, cookies will be sent only to hosts in that domain. By default, the cookie will only be sent to the host from which it was generated.

IBI_COOKIE_PATH. The path(s) on the server to which the cookie will be sent. By default, WebFOCUS sets the path to the root directory (“/”).

IBI_COOKIE_SECURE. When enabled, this variable restricts the exchange of cookies to only HTTPS sessions. This variable is not set by default.

Reference **WebFOCUS Cookies and Parameters**

WebFOCUS Cookie	Parameters/Descriptions
WF_COOKIE	<ul style="list-style-type: none"> • IBIC_user - WebFOCUS Reporting Server user ID. • IBIC_pass - WebFOCUS Reporting Server user password. • IBIC_server - WebFOCUS Reporting Server name. • IBID_user - Alternate Deferred Receipt Server user ID. • IBID_pass - Alternate Deferred Receipt Server user password. • IBID_server - Alternate Deferred Receipt Server name. <p>These optional parameters are used for connection purposes only. They are not available to the WebFOCUS Reporting Server, and they will only be saved in WF_COOKIE if they appear on a Web page that uses a WF_SIGNON action.</p>
WF_USER	Includes any optional variables other than those listed for WF_COOKIE that are found on the signon request.
WF_VIEWER	For internal use only. These optional parameters are required for on-demand paging and cookie expiration.
MR_COOKIE	Contains information about the MRE session, including the MRE user ID of the authenticated user. For more information about MR_COOKIE, see <i>MRE Signon Processing</i> on page 3-14.
MR_INFO	Contains additional internal MRE information.

Controlling Cookie Expiration

By default, if the WebFOCUS cookie is set in a browser, it expires when the browser is closed. However, an expiration value can be set in the `ibidir.wfs` file. This expiration time is refreshed each time a WebFOCUS report is run. The cookie is deleted when the expiration time is reached. Once a cookie is deleted, user credentials must be reentered in order to continue to use WebFOCUS.

Syntax

How to Control Cookie Expiration

To set an expiration time, edit the following syntax in the `ibidir.wfs` file

```
WF_COOKIE_EXPIRATION=time
```

where:

time

Is the time in minutes that you want the current user credentials to remain in effect. The default time is 0 minutes, meaning that the cookie expires when the browser closes.

Encrypting the WebFOCUS User Cookie

To encrypt the WebFOCUS user cookie, set the following flag (in `ibiweb.cfg`):

```
ENCRYPT_WF_USER=yes
```

Note: The `WF_USER` cookie is not encrypted by default.

Integrating Web Server Security With WebFOCUS

It may be advantageous to integrate Web server security with WebFOCUS. This is enabled using the Web server HTTP variable `&REMOTE_USER`, which stores the value of the authenticated user's logon ID. This variable is populated only if Web server authentication is enabled and will be empty if anonymous authentication is in effect. This variable can be exported to the WebFOCUS application environment as a Dialogue Manager variable using the `<sendvar>` command in the `site.wfs` file.

To take advantage of this feature, place the following text in your `site.wfs` file:

```
<sendvar>  
WEBAUTH = &REMOTE_USER  
<endsendvar>
```

or

```
<sendvar>  
WEBID = &WF_REMOTE_USER  
<endsendvar>
```

The `WEBAUTH` and `WEBID` Dialogue Manager variables will be available to your report procedures. Select any name you like for the variables. In the NT environment, `&REMOTE_USER` is populated with the Windows NT/2000 domain name. `&WF_REMOTE_USER` includes the user ID and strips the domain name away for you.

These variables cannot be used in the WebFOCUS Reporting Server's global profile. However, they are available to a WebFOCUS site profile, which you can create in order to achieve a similar effect. For more information, see Chapter 6, *Tips and Techniques*.

Note: Always test your environment to determine its actual behavior. If the Web server is on a Windows NT/2000 PDC, the domain is generally not included in `&REMOTE_USER` for domain users. If the Web server is on a stand-alone server in the domain, `&REMOTE_USER` will be populated as DOMAIN/user ID.

See Chapter 6, *Tips and Techniques*, for an example of *Application Security Integration*.

WebFOCUS and Network Layer Security

When network layer security is implemented, it is usually outside the scope of WebFOCUS. However, for the WebFOCUS Client to talk to the WebFOCUS Reporting Server, the Web server IP address must be able to send TCP traffic through the firewall or proxy to the port and address where the WebFOCUS Reporting Server is running.

WebFOCUS and Data Source Security Integration

Many data sources offer an opportunity for a trusted relationship with WebFOCUS. For example, DB2 on OS/390 supports this with a special exit that can be installed. Microsoft SQL Server and Oracle® also offer this by way of different mechanisms. Refer to your iWay Server Administration documentation for specifics about your data source.

Managed Reporting Environment (MRE) Security

The WebFOCUS Managed Reporting Environment (MRE) makes it easy to deploy a reporting environment that balances the flexibility and power demanded by users with the security and manageability sought by administrators.

Using a built-in administrator interface, the MRE Administrator can quickly build an environment that manages user access to reporting resources based on groups, reporting domains, and user capabilities. There is also a programming exit that can be called to extend the functionality and scalability of the MRE. For more information, see Chapter 4, *WebFOCUS Custom Security Exits*.

The following sections describe MRE security.

Built-In Security Features

MRE offers an “out-of-the-box” solution for deploying a secure reporting environment. Key features related to the management of MRE security are:

MRE Administrator. One or more MRE users can be granted MRE Administrator rights. These users have the ability to create and manage user accounts, reporting domains, and user groups. The default MRE Administrator account created during installation is admin and has no initial password. It is recommended that you grant this account a password or disable the account.

MRE Domain Administrator. These users can manage the domains they have access to. They have many of the capabilities of MRE Administrators except that they cannot create and manage user accounts, and they cannot create new domains or assign groups to domains.

MRE Users. After their account has been defined by the MRE Administrator and they have logged on, users can access designated reporting resources. There are three different kinds of users: Java applet users, HTML-based users, and Windows version users.

Reporting Domains and User Groups. Administrators can create reports and report templates (also known as reporting objects) and organize them in reporting domains. These domains can then be mapped by an MRE Administrator to one or more user groups. Based on each user’s group membership, MRE ensures that they can only access the domains they are authorized for.

User Capabilities. In addition to the user domain mapping managed by MRE, the MRE Administrator can also control which MRE features are available to each user. The following user capabilities can be controlled:

- Type of user account - Java applet, HTML, or Developer Studio-based.
- Ad-hoc enabled - whether a Java user can create their own reports.
- Run only user - these users cannot save reports to MRE.
- Password - user passwords can be set or overridden by the MRE Administrator.
- Shared Report access - whether the user can share their own reports with other users.
- Schedule capability - the user can access the optional ReportCaster feature.
- FML capability - the Developer Studio user can use the Financial Markup Language feature.
- PDA Sync - the user can access the optional PDA Sync mobile feature.

MRE Signon Processing

The standard MRE logon forms (logon.htm, logons.htm, logonsd.htm) forms can be copied and modified to suit the individual site's needs. MRE signon customization techniques and customized programming exits can also be employed. For more information, see *MRE Signon Integration* on page 3-16 and Chapter 4, *WebFOCUS Custom Security Exits*.

Regardless of which logon form is used, the following HTML variables are set to log on to MRE:

```
<INPUT TYPE="HIDDEN" NAME="IBIMR_action" VALUE="MR_SIGNON">  
<INPUT TYPE="HIDDEN" NAME="IBIMR_random" VALUE="">
```

The IBIMR_action command instructs WebFOCUS to authenticate both MRE and WebFOCUS Reporting Server user credentials. If the user has not supplied WebFOCUS Reporting Server user credentials on the form, the default settings in the ibidir.wfs file are validated instead.

The IBIMR_random="" command, together with a special JavaScript function on the page, is used to defeat browser caching algorithms that can cause signon problems.

MRE Cookie (MR_COOKIE). If the signon is successful, the MRE cookie (MR_COOKIE) is set. This cookie contains the MRE user ID of the authenticated user. The MRE user password is not stored in this cookie. The fact that the MRE user ID was set in the cookie is sufficient to indicate that authentication has taken place, so the MRE does not need the password for any subsequent requests. MR_COOKIE is always encrypted, and it expires at the end of a session or when an MRE log off occurs.

The WebFOCUS cookie (WF_COOKIE) is not set by the IBIMR_action command. It is only set if the IBIWF_action=WF_SIGNON command is processed, which depends on which MRE signon form is used. For more information, see *WebFOCUS Cookies* on page 3-9.

Reference

Standard MRE Signon Forms

The following MRE signon forms are located in ibi_html/workbnch/:

- logon.htm - Managed Reporting Signon only.
- logons.htm - Managed Reporting and WebFOCUS Reporting Server Signon.
- logonsd.htm - Managed Reporting, WebFOCUS Reporting Server, and Alternate Deferred Server Signon.

By default, mrlogon.htm is a frameset which calls mrlogon1.htm. This in turn calls logons.htm. These pages can be copied and substituted as needed. The key HTML tags on each form are as follows:

Form	Significant HTML Form Tags
logon.htm	<pre><INPUT TYPE="TEXT" NAME="IBIMR_user" VALUE="" > <INPUT TYPE="PASSWORD" NAME="IBIMR_pass" VALUE="" > <INPUT TYPE="HIDDEN" NAME="IBIMR_action" VALUE="MR_SIGNON"> <INPUT TYPE="HIDDEN" NAME="IBIMR_random" VALUE="" ></pre>
logons.htm	<pre><INPUT TYPE="TEXT" NAME="IBIMR_user" VALUE="" > <INPUT TYPE="PASSWORD" NAME="IBIMR_pass" VALUE="" > <INPUT TYPE="TEXT" NAME="IBIC_user" VALUE="" > <INPUT TYPE="PASSWORD" NAME="IBIC_pass" VALUE="" > <INPUT TYPE="HIDDEN" NAME="IBIWF_action" VALUE="WF_SIGNON"> <INPUT TYPE="HIDDEN" NAME="IBIMR_action" VALUE="MR_SIGNON"> <INPUT TYPE="HIDDEN" NAME="IBIMR_random" VALUE="" ></pre>
logonsd.htm	<pre><INPUT TYPE="TEXT" NAME="IBIMR_user" VALUE="" > <INPUT TYPE="PASSWORD" NAME="IBIMR_pass" VALUE="" > <INPUT TYPE="TEXT" NAME="IBIC_user" VALUE="" > <INPUT TYPE="PASSWORD" NAME="IBIC_pass" VALUE="" > <INPUT TYPE="TEXT" NAME="IBID_user" VALUE="" > <INPUT TYPE="PASSWORD" NAME="IBID_pass" VALUE="" > <INPUT TYPE="HIDDEN" NAME="IBIWF_action" VALUE="WF_SIGNON"> <INPUT TYPE="HIDDEN" NAME="IBIMR_action" VALUE="MR_SIGNON"> <INPUT TYPE="HIDDEN" NAME="IBIMR_random" VALUE="" ></pre>

MRE Signon Integration

It is possible to tailor the MRE signon process to suit various objectives. For example, you may want to link an MRE user ID with a WebFOCUS Reporting Server user ID, thereby avoiding prompting for both.

Other considerations might be to integrate MRE signon with Lightweight Directory Access Protocol (LDAP), Web server authentication, or a home-grown security system.

Example

Integrating MRE and WebFOCUS Reporting Server Signons

It is possible to prompt for and validate a WebFOCUS Reporting Server user ID and sign into MRE with that user ID. The following commands would be placed in the site.wfs file:

```
IBIMR_user = &IBIC_user  
IBIMR_pass = &IBIC_pass
```

Typically, a user's WebFOCUS Reporting Server password expires on a set schedule. In this case, there will be a problem with the above example, since the passwords for the WebFOCUS Reporting Server and MRE will eventually lose synchronization. To address this issue, the following commands would be more appropriate:

```
IBIMR_user = &IBIC_user  
IBIMR_pass =
```

With this example, all MRE user accounts have the same fixed password ("null" in the example above) and the synchronization issue is avoided. Sharing a fixed password across all MRE accounts is not necessarily a security exposure. Simply knowing the value of the password would not allow you to log on to someone else's MRE account. You can only log on to your own account because the IBIMR_user value is set in the command.

Note: During MRE signon processing, once WebFOCUS Reporting Server credentials have been validated, if no value for IBIMR_user and IBIMR_pass is found anywhere, the values for IBIC_user and IBIC_pass will be validated by MRE instead. However, the recommended approach in synchronizing these user IDs is to use the commands shown above.

Example

Integrating MRE and Web Server Signons

When Web server authentication is configured, it is common to link the Web server and MRE user IDs. Do this by placing the following commands in the site.wfs file:

```
IBIMR_user = &WF_REMOTE_USER  
IBIMR_pass =
```

Create the MRE accounts with no password, as in the previous example.

Reference

Making the Signon Integration Conditional

You may want to use one of the previous integration examples for most users, but allow certain users to supply an MRE user ID and password at signon. For example, check if the `IBIMR_user` value passed at signon is 'admin'. If so, validate it with the value passed for `IBIMR_password`. Otherwise, ignore any such values and use the integrated configuration.

You can accomplish this using the following commands:

```
<if> IBIMR_user NE "admin"
IBIMR_user = &WF_REMOTE_USER
IBIMR_pass =
<endif>
```

If you are using WebFOCUS Version 4 Release 3.5 or higher, you should substitute the following code for the first line in the example above to avoid case-sensitivity problems:

```
<if> IBIMR_user.upper NE "ADMIN"
```

Considering the conditional wrapper syntax is particularly important if you are using ReportCaster against a secure Web server. For example, if the `HTTPUSER` parameter in `bkrsched.cfg` is 'user1' and `MREUSER` is 'admin' you would not be able to retrieve MRE procedures unless you implemented the conditional syntax shown above.

MRE Internal Variables

During the MRE signon event, WebFOCUS stores the authenticated MRE user ID in the `MR_COOKIE`. It is possible to export this value to the WebFOCUS Reporting Server environment for use in a report procedure. For example, you could print the MRE user ID in the heading of the report.

In addition, you can use the `IBIMR_domain` internal variable to expose the currently opened MRE domain to which the user is granted access.

To accomplish this, use `<SET>` statements to export the internal variables used to store the MRE user ID and domain to a WebFOCUS variable of your choice.

The MRE internal variable `IBIMR_user` can be accessed anywhere inside a procedure that executes on the WebFOCUS Reporting Server. However, you must tell the WebFOCUS Client to ignore this variable when parsing the procedure in MRE or WebFOCUS will prompt the user to enter a value for `MREuser`. You can accomplish this by either unchecking the Prompt for Parameters report property, or by using the `MRNOEDIT` syntax. For more information about `MRNOEDIT`, see the *WebFOCUS Managed Reporting Administrator's Manual*.

Note: You may want to use the internal variable *IBIMR_user* in the WebFOCUS Reporting Server's profile (edasprof.prf). Although variables are not available for use in this profile, there is a technique that can be used to achieve similar results. For more information, see *Creating WebFOCUS Site Profiles*, in Chapter 6, *Tips and Techniques*.

Syntax

How to Code MRE Internal Variables

The following syntax must be placed at the bottom of the site.wfs file

```
<SET> IBIMR_user(PASS)
<SET> IBIMR_domain(PASS)
```

where:

IBIMR_domain

Is the name of the Dialogue Manager variable for the current MRE domain. The current MRE domain is set whenever you run or view something in Domains or Domain Builder. However, there is no current MRE domain when you use the Deferred Receipt and Data Server features, or when you are building a Reporting Object. *IBIMR_user*

Is the name of the Dialogue Manager variable for the authenticated MRE user ID.

Example

Running MRE Internal Variables

Running the MRE internal variable code with a -TYPE statement can produce the following results:

```
IBIMR_user = admin
IBIMR_domain = domain/domain.htm
```

Note: Uncheck the Prompt for Parameters report property when saving the request of type statements.

Securing the MRE Repository

MRE uses a directory structure on disk to store information related to the reporting application. This storage structure is called the MRE Repository and is usually located on the Web server platform.

The MRE Repository contains report code, metadata about the reporting environment, support files (such as images and HTML), and finished report output (such as a PDF or Excel file). It is important to protect this information from unauthorized access.

During normal operation, the MRE Repository should only be accessed by the WebFOCUS Client, which manages access to MRE Repository information. Depending on file permissions, it may be possible for users to circumvent WebFOCUS security and view or change sensitive MRE Repository information by using operating system features such as telnet and vi. This situation should be avoided because sensitive information can be exposed, production reports can be altered, and MRE itself can be rendered inoperable. When this exposure has been eliminated, the site is considered to have implemented a secure MRE Repository.

The basic concept of a secure MRE Repository is simple: the effective ID of the WebFOCUS Client program is controlled so that it is the same for all user connections. MRE Repository file system permissions are changed so that only this fixed ID is allowed access to the MRE Repository. An administrator group can also be configured with file system access to the MRE Repository for the purpose of copying files to its import directories, or for support and debugging purposes.

For details on how to secure your MRE Repository, see the *WebFOCUS and ReportCaster Installation and Configuration* manual for your platform.

Deferred Receipt Security

Deferred Receipt is an MRE feature that allows users to submit an MRE procedure that executes in the background. The user then views the finished report output from the Deferred Report Status interface in MRE. This is in contrast to procedures submitted for immediate execution, where the browser waits for the request to finish.

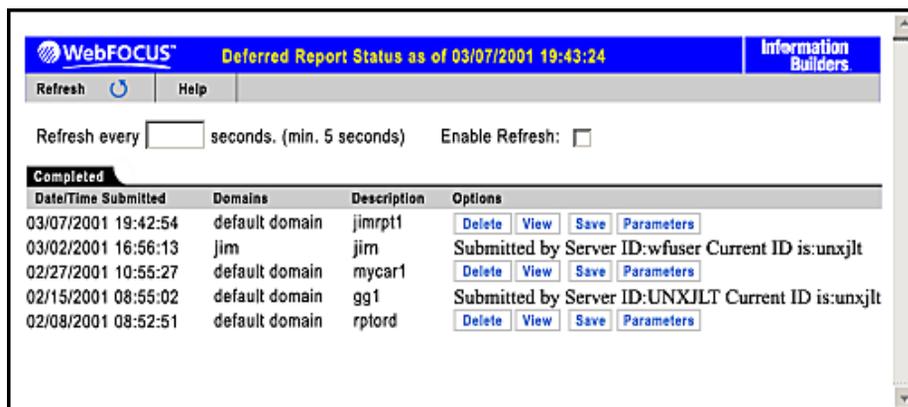
From a security perspective, deferred requests are accepted by the WebFOCUS Reporting Server in the same way as immediate requests. If WebFOCUS Reporting Server security is enabled, the deferred request must connect with a valid user ID and password. This user ID may be the user's WebFOCUS signon ID (IBIC_user) or an alternate deferred server signon ID (IBID_user).

When the request has been run, by default the output is stored in a file on the WebFOCUS Reporting Server under the *install_directory/srv436/conf/dfm_dir* with a corresponding file that contains the user ID and other information. The WebFOCUS Reporting Server ensures that only the user who has submitted the deferred job can retrieve and delete the output file. The WebFOCUS Reporting Server Administrator (the user identified by the *server_admin_id* keyword in the *edaserve.cfg* file) can also view and delete any deferred output, but can do so only at the file level or by using the iWay Server Console. To delete a deferred job, the administrator would need to know its ticket value.

Access to *dfm_dir* should be restricted so that the user ID that started the WebFOCUS Reporting Server has read/write access. Read access should be controlled so that unauthorized users cannot gain access to the directory.

From the MRE perspective, a deferred ticket is stored in the MRE Repository for each user's deferred request. The tickets are stored by MRE user, and each user can only see their own deferred tickets.

The following displays a Deferred Report Status window:



In this display, the user has logged in with the MRE Administrator account using WebFOCUS Reporting Server user ID unxjlt.

- What appears on the list of deferred jobs is controlled by the MRE user ID, which is not case sensitive.
- Which deferred jobs can be viewed depends on the user's WebFOCUS Reporting Server user ID, which is case sensitive.

The job run on 03/02/2001 cannot be viewed because it was submitted during a session where the user had been using the default connect user ID specified as wfuser in ibidir.wfs. The other jobs have all been run from sessions where a WebFOCUS signon took place, overriding the default in ibidir.wfs. However, the job run on 2/15/2001 cannot be viewed because during that WebFOCUS signon the user ID was specified in uppercase. Be sure to consider the effect of allowing both an ibidir.wfs default and a WebFOCUS signon capability in your WebFOCUS implementation.

ReportCaster Security

ReportCaster is a Java language-based scheduling and distribution application that centralizes the execution and distribution of WebFOCUS reports over the Web. ReportCaster provides a single point of control and administration to manage the many reports required to run an organization.

A properly configured and managed implementation of ReportCaster should provide an acceptable level of security for any organization. For detailed information about ReportCaster security, see the *WebFOCUS ReportCaster Administrator's Manual*.

CHAPTER 4

WebFOCUS Custom Security Exits

Topics:

- WebFOCUS CGI Callable Exits
- WebFOCUS Servlet Callable Exits
- MRE Callable Exits
- WebFOCUS DBA Exits
- Pre-Verify User ID Exit

If you want to customize the WebFOCUS Client or WebFOCUS Reporting Server, user written security exits are available. These exits may be used to integrate WebFOCUS with a Lightweight Directory Access Protocol (LDAP) server. Customers with special security requirements can use one of the following types of custom security exits:

- WebFOCUS Client Exits:
 - WebFOCUS CGI Callable Exits (WFEXT module).
 - WebFOCUS Servlet Callable Exits (WFEXT module).
 - MRE Callable Exits (MRCEXT module).
- WebFOCUS Reporting Server Exits:
 - WebFOCUS DBA Exits (XDBSC module).
 - Pre-Verify User ID Exits (PVUIDXT module).

WebFOCUS CGI Callable Exits

WebFOCUS callable exits are part of the WFEXT module. They enable WebFOCUS developers to customize the execution of the Communications Gateway Interface (CGI) for better integration with their application environment (for example, LDAP). Developers can write up to 20 WebFOCUS CGI callable exits within the WFEXT module. While generally aimed at the self-service user, this feature can also be used in the Managed Reporting Environment (MRE).

The WebFOCUS Client accepts variables that the WebFOCUS CGI callable exit places in an output buffer. The output buffer contains a list of name/values pairs delimited by an ampersand (&). After the WFEXT program is executed, the WebFOCUS Client parses the output buffer to accept passed parameters. The number of parameters in the output buffer depends on the number of variables the developer expects to pass to the WebFOCUS Client. For details, see *How to Define a WebFOCUS CGI Callable Exit* on page 4-3. The name of the WebFOCUS CGI callable exit is set in *retfuncs*. For more information about *retfuncs*, see *Retfuncs* on page 4-5.

The WebFOCUS CGI callable exits can be written in C or in another language that supports the creation of shared objects or dynamic link libraries. For servlets, the routines can be written in the Java language. For more information about servlet callable exits, see *WebFOCUS Servlet Callable Exits* on page 4-12.

Reference

Typical Implementation of WebFOCUS CGI Callable Exits

One way to use this exit is with an external security repository, like LDAP, to garner user profile information that is not provided by the user. For example, suppose that a user needs a group ID to connect to OS/390, and a different group ID to connect to UNIX. By creating a WebFOCUS CGI callable exit within WFEXT, a developer can look into a repository and gather the group IDs as connection information (as well as other parameters such as employee ID and SSN). These parameters can then be passed to the WebFOCUS Reporting Server for standard amper variable processing.

To enable a WebFOCUS CGI callable exit, you must set position 10 of *MR_PGM_FLAGS* to Y. For more information about *MR_PGM_FLAGS*, see Chapter 5, *WebFOCUS Administration*. The WebFOCUS CGI callable exit is invoked by using a *<CALL>* statement. For more information, see *Invoking a WebFOCUS CGI Callable Exit* on page 4-11.

Syntax**How to Define a WebFOCUS CGI Callable Exit**

Include the following code in the WFEXT module

```
status funcname (NewVars, parm1, parm2, . . . parm10)
```

where:

status

Is the status value the function can return. The data type is long, and the status value is determined by the creator of the routine. Status is passed to the WebFOCUS script (WFS) variable *retcode*. A status of 999 is returned when the function is not found.

funcname

Defines the name used in the WFS <CALL> statement (for example, *security*).

NewVars

Is the buffer where new or updated variables are to be returned. The maximum return buffer is 8K. The NewVars format is as follows:

```
name1=value1&name2=value2
```

where:

name1, name2 are the names of variables.

value1, value2 are the values associated with each variable.

& separates variable values from the next variable name.

parm1, parm2, . . . parm10

Are the input parameters of the WebFOCUS CGI callable exit. Each exit can contain a maximum of 10 input parameters. The number of parameters following the output buffer varies, as determined by the number of variables the developer expects to pass to the exit.

WebFOCUS Reporting Server Input Processing (TRANSIN)

The WebFOCUS Reporting Server Input Processing (*transin*) exit function is called right before the procedure is passed to the WebFOCUS Reporting Server. The developer can then modify the procedure (*inbuf*) as necessary. For more information, see *How to Use WebFOCUS Reporting Server Input Processing* on page 4-4.

The following syntax in *retfuncs* activates and automatically calls WebFOCUS Reporting Server Input Processing:

```
userfunc->userinpext=(WF_INPUTEXIT *)transin;
```

For more information about *retfuncs*, see *Retfuncs* on page 4-5.

Syntax

How to Use WebFOCUS Reporting Server Input Processing

```
transin(char *inbuf, char *outbuf, long inbytes, long *outbytes,  
        long *userflag)
```

where:

transin

Is the name of the exit function.

inbuf

Is the input buffer.

outbuf

Is the output buffer.

inbytes

Is the length of the input buffer.

outbytes

Is the length of the output buffer.

long

Is the status value that the function can return.

userflag

Is initially set to 0 by the CGI.

WebFOCUS Reporting Server Output Processing (TRANSOUT)

The WebFOCUS Reporting Server Output Processing (transout) exit function behaves in the same manner as the Input function. It is called for each report line the user receives from the WebFOCUS Reporting Server. An additional parameter called *type* allows the developer to determine the format of the output.

The following syntax in retfuncs activates and automatically calls WebFOCUS Reporting Server Output Processing:

```
userfunc->useroutext=(WF_OUTPUTEXIT *)transout;
```

For more information about retfuncs, see *Retfuncs* on page 4-5.

Syntax

How to Use WebFOCUS Reporting Server Output Processing

```
transout(char *inbuf, char *outbuf, long inbytes, long *outbytes,
         char *type, long *userflag)
```

where:

transout

Is the name of the exit function.

inbuf

Is the input buffer.

outbuf

Is the output buffer.

inbytes

Is the length of the input buffer.

outbytes

Is the length of the output buffer.

type

Allows the user to determine the format of the output.

long

Is the status value that the function can return.

userflag

Is initially set to 0 by the CGI.

Retfuncs

Retfuncs is the function that determines which WebFOCUS CGI callable exits will be called by WFEXT. It defines the number of WebFOCUS CGI callable exits.

In addition, retfuncs can also call the WebFOCUS Reporting Server Input Processing (transin) exit function:

```
userfunc->useroutext=(WF_INPUTTEXT *)transin;
```

Retfuncs can also call the WebFOCUS Reporting Server Output Processing (transout) exit function:

```
userfunc->useroutext=(WF_OUTPUTTEXT *)transout;
```

Syntax**How to Use Retfuncs to Define WebFOCUS CGI Callable Exits to the WebFOCUS Client**

The following syntax can be used in retfuncs to define WebFOCUS CGI callable exits to the WebFOCUS Client

```
retfuncs(t_exits *userfunc,t_rout *urout)
```

where:

userfunc

Automatically invokes the WebFOCUS Reporting Server Input (transin) and WebFOCUS Reporting Server Output (transout) exit functions.

urout

Defines the internal table that becomes populated with WebFOCUS CGI callable exits. The code for *urout* is:

```
long num_of_func=x;
char funcname[y];
FUNCPTR *function
```

where:

x defines the number of WebFOCUS CGI callable exits, excluding transin and transout.

funcname defines the name used in the WFS <CALL> statement.

y defines the position of the WebFOCUS CGI callable exit in the table.

function defines the name of the actual function to be called.

Example**Populating WFEXT With Retfuncs**

Retfuncs determines which WebFOCUS CGI callable exits WFEXT calls by populating an internal table that can contain up to 20 exits. The following is an example of what this table can look like (where $x=5$).

funcname from <CALL> statement	y	actual name of function
security	0	security
security1	1	security1
security2	2	security2
security3	3	security3
security4	4	security4

For an example of a WebFOCUS CGI callable exit program, see *Creating a WebFOCUS CGI Callable Exit* on page 4-7.

Example **Creating a WebFOCUS CGI Callable Exit**

The following is a sample WebFOCUS CGI callable exit that is used to look up a user's security credentials from an external source. Although this exit uses an input file, it could be modified to use LDAP or a DBMS.

Note: A C header file is needed when coding the exit.

```

/*-----
 * Copyright (c) 2000 Information Builders, Inc. All rights reserved.
 *
 * _Name_          ==> WFCEXIT  H          _Opsys_      ==> CMS
 * _              _Product_    ==> FOC
 *
 * DEFINITION OF RETFUNCS
 *-----
 */
typedef void WF_INPUTEXIT(char *,char *,long,long *,long *);
typedef void WF_OUTPUTEXIT(char *,char *,long,long *,char *,long *);

typedef long FUNCPTR(char *,...);

typedef struct s_userout {
    char name[20];
    FUNCPTR *function;
} t_userout;

typedef struct s_rout{
    long num_of_func;
    t_userout ur[20];
} t_rout;

typedef struct s_exits {
    WF_INPUTEXIT *userinext;
    WF_OUTPUTEXIT *useroutext;
} t_exits;

typedef void WF_RETFUNCS(t_exits *,t_rout *);

```

```

/*
*****
*
* Program ID: wfext.c
* Module ID : WinNT= wfext.dll
*           UNIX = libwfext.so or libwfext.sl
*
* This code is sample code, provided as-is.
* No warranties are made as to its correctness or
* suitability for any purpose.
*
* Description: Sample X-PLATFORM WebFOCUS User Written exit
*
* Notes      : To activate the wfext exit, position 10 of
*             MR PGM FLAGS must be set to Y.
*
*****
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <errno.h>
#include "wfcexit.h"

#define WF_SUCCESS      0
#define WF_FAIL_PARM   1
#define WF_FAIL_NOFILE 2
#define WF_FAIL_NOTFOUND 3
#define SECFILE "wfext.dat"
/*
*****
* WebFOCUS user callable routine
*
* Function -
* Name:
*   security
*
* Input File:
*   wfext.dat - External coma delimited file of user information
*             format: key,eda server,user id,password,e-mail address
*             example: alan,EDASERVE,USR0001,SECRET,usr0001@ibi.com
*
* Input Parameters:
*   pszUser - lookup key
* Output Buffer:
*   NewVars - Buffer (max 8k) to return new or updated variables,
*             Format: name=value&name=value,...
* Return Codes:
*   0=SUCCESS
*   1=Input key is missing
*   2=External lookup table not found
*   3=Lookup entry not found

```

```

*****
*/
long security(char *NewVars, char *pszUser)
{
    FILE      *fp;
    long      rc=WF_SUCCESS;
    int       iUser,
             nParm,
             iSiz,
             iLen;
    char      *pszKey,
             sep[]="\n\r\0",
             *pszValue,
             szTable[120];

    /* ----- */
    /* format of external table is */
    /* {key,parmvalue1,parmvalue2,...} */
    /* ----- */
    /* layout: key,server,user,password,e-mail addr */
    /* defines variable name assignments, key is not */
    /* part of assignmets. */
    /* ----- */
    char *pszName[]={ "wfenode", "wfeuser", "wfepass", "wfemail", NULL };

    /* ----- */
    /* make sure input parameter & security file exist */
    /* ----- */
    if (pszUser==NULL)
        return(WF_FAIL_PARM);

    if ((fp = fopen(SECFILE, "r")) == NULL)
        return(WF_FAIL_NOFILE);

    /* ----- */
    /* File lookup: */
    /* search for matching key */
    /* ----- */
    while (fgets(szTable, 120, fp))
    {
        iLen=strlen(szTable);
        if (iLen>1)
        {
            /* get key */
            pszKey=strtok(szTable, sep);
            if (pszKey!=NULL)
            {
                /* match key */
                iUser=strcmp(pszKey, pszUser);
                if (iUser==0)
                {
                    /* OK: key match */
                    rc=WF_SUCCESS;
                    nParm=0;
                }
            }
        }
    }
}

```

```

        iLen=0;
        while (pszName[nParm])
        { /* format output buffer, name=value */
            pszValue=strtok(NULL,sep);
            if (pszValue==NULL)
                break;

iSiz=sprintf(NewVars+iLen,"%s=%s&",pszName[nParm],pszValue);
            iLen+=iSiz;
            nParm++;
        }
        *(NewVars+iLen-1)='\0'; /* remove trailing & from output buffer*/
        break;
    }
    else
        rc=WF_FAIL_NOTFOUND;
}
}
fclose(fp);
return(rc);
}
/*
*****
* WebFOCUS pre-defined user written exit: INPUT PROCESSING
* To activate uncomment "userfunc->userinpxt=(WF_INPUTTEXTIT *)transin;" in
retfuncs
*****
*/
void transin(char *inbuf,char *outbuf,long inbytes,long *outbytes,
            long *userflag)
{
    memcpy(outbuf,inbuf,inbytes);
    *outbytes=inbytes;
}
/*
*****
* WebFOCUS pre-defined user written exit: OUTPUT PROCESSING
* outbuf will be allocated by the caller to be 2*inbytes incase of conversion to DBCS
* type will contain format of the output .htm,.html,.pdf etc.
* To activate uncomment "userfunc->useroutext=(WF_OUTPUTTEXTIT *)transout;" in retfuncs
*****
*/
void transout(char *inbuf,char *outbuf,long inbytes,long *outbytes,
            char *type,long *userflag)
{
    memcpy(outbuf,inbuf,inbytes);
    *outbytes=inbytes;
}
*
/*
*****
* WebFOCUS defined: Retfuncs (used by CGI to determine which exits to call)

```

```

*****
*/
void
#ifdef WIN32
__declspec( dllexport )
#endif
retfuncs(t_exits *userfunc,t_rout *urout)
{
    /* Uncomment to use transin and transout:
    /* userfunc->userinpevt=(WF_INPUTEXIT *)transin; */
    /* userfunc->useroutext=(WF_OUTPUTEXIT *)transout; */

    /* Define WFS user callable routine to CGI */
    /* number of callable routines (maximum 20) */
    urout->num_of_func=1;
    /* name used in the WFS<CALL> statement */
    strcpy(urout->ur[0].name,"security");
    /* Actual function to be called */
    urout->ur[0].function=(FUNCPTR *)security;
}

```

Example**Invoking a WebFOCUS CGI Callable Exit**

Once the WebFOCUS CGI callable exit is created, add the following code to site.wfs (or ibidir.wfs) to invoke the exit:

```

# $Revision: 1.12 $
<! site.wfs setup file>
<VER 1>

<! ----->
<! get secur. cred. from an external source >
<! ----->
<CALL> security(seccode)
<if> RETCODE EQ "0"
IBIC_pass = &wfepass
IBIC_user = &wfeuser
IBIC_server = &wfenode
<else>
<EXIT>

```

Compiling a WebFOCUS CGI Callable Exit

The following is a sample make file for compiling a WebFOCUS CGI callable exit on a Sun Microsystems platform:

```
SHELL    = /bin/ksh

DEFINES=-D_REENTRANT -DFT_UNIX
CFLAGS=-g ${DEFINES}
LIBS=-G -lc

all: wfcexit

clean:
    rm -f wfcexit.o

wfcexit: wfcexit.o
    cc -o libwfext.so wfcexit.o ${LIBS}
```

The following is a sample for compiling a WebFOCUS CGI callable exit on an IBM Open Edition UNIX Systems Services (OS/390) platform:

```
cc -o libwfext.so -W c,dll,expo -W l,dll wfcexit.c
```

Note: For more information about make files, see your operating system documentation.

WebFOCUS Servlet Callable Exits

WebFOCUS servlet callable exits are part of the WFEXT module. They enable WebFOCUS developers to customize the execution of the servlet for better integration with their application environment (for example, LDAP). There is no limit to the number of methods that may be defined and used within the WFEXT module.

Procedure How to Configure a WebFOCUS Servlet Callable Exit That Uses a Legacy Servlet Container

Modify your Java servlet configuration by:

1. Adding the following initialization arguments

```
Init Parm Name=WFEXT
Init Parm Value=classname
```

where:

classname

Can be any class name you choose. This is the name of the class that contains the methods you will be calling with the <CALL> method name.

These arguments are added so that the WebFOCUS servlet knows to load your class. Otherwise, the servlet will not load your class and the WebFOCUS servlet callable exit will not work.

2. Adding the exit's class to the classpath. This is used to locate your class file.

Note:

- The MR_PGM_FLAGS setting is not necessary for this exit.
- Retfuncs is not used (or required) to enable this exit.

Procedure How to Configure a WebFOCUS Servlet Callable Exit That Uses a J2EE Web Application

Modify your Java servlet configuration by:

1. Adding the following initialization arguments in the web.xml file

```
<init-param id="InitParam_2">
<param-name>WFEXT</param-name>
<param-value>classname</param-value>
<description>WebFOCUS servlet callable exit</description>
</init-param>
```

where:

classname

Can be any class name you choose. This is the name of the class that contains the methods you will be calling with the <CALL> method name.

These arguments are added so that the WebFOCUS servlet knows to load your class. Otherwise, the servlet will not load your class and the WebFOCUS servlet callable exit will not work.

2. Adding the exit's class to the web-inf/classes directory of the Web application.

Note:

- The MR_PGM_FLAGS setting is not necessary for this exit.
- Retfuncs is not used (or required) to enable this exit.

Syntax

How to Define a WebFOCUS Servlet Callable Exit

Include the following code in the WFEXT module

```
public long routinename(String[] NewVars, HttpServletRequest req, String var1)
```

where:

NewVars

Is the buffer where new or updated variables are to be returned. The NewVars format is as follows:

```
name1=value1&name2=value2
```

where:

name1, *name2* are the names of variables.

value1, *value2* are the values associated with each variable.

& separates variable values from the next variable name.

req

Is used to integrate parameters in the HTTP request buffer (for example, sessions).

var1

Defines the number of input parameters. This is an optional parameter.

Example**Creating a WebFOCUS Servlet Callable Exit**

The following is a sample WebFOCUS servlet callable exit that retrieves the authenticated user ID, password, and server node from a session object. Name/value pairs are then inserted into NewVars. Variables that are needed for reporting purposes may be sent to the WebFOCUS Reporting Server using the SENDVAR or SET PASS commands.

```
public class WfCallableExit {
/**
 * WfCallableExit constructor comment.
 */
public WfCallableExit() {
    super();
}
public long getSession(String[] NewVars, HttpServletRequest req)
{
    String Node;
    String Userid;
    String Password;
    try
    {
        HttpSession session = req.getSession(false);

        Node = (String) session.getAttribute("node");
        if (Node == null)
        {
            NewVars[0]=setNewVars("?", "?", "?");
            return 1;
        }
        Userid = (String) session.getAttribute("userid");
        Password = (String) session.getAttribute("password");

        NewVars[0]=setNewVars(Userid, Password, Node);
        return 0;
    }
    catch (NullPointerException e)
    {
        NewVars[0]=setNewVars("?", "?", "?");
        return 2;
    }
}
/**
 * Populates WebFOCUS internal exit buffer with name/value pairs the & is
not part
 * of the variable, it is used as a delimiter.
 */
private String setNewVars(String user, String pswd, String server)
{
    String NewVars;
    NewVars = "user=" + user + "&pswd=" + pswd + "&server=" + server;
    return(NewVars);
}
}
```

Example Invoking the WebFOCUS Servlet Callable Exit

Once the WebFOCUS servlet callable exit is created, the following code should be added to site.wfs (or ibidir.wfs) to invoke the exit:

```
<call> getSession()
<if> RETCODE EQ "0"
    IBIC_user=&user
    IBIC_pass=&pswd
    IBIC_server=&server
<else>
    <HTMLFORM> wfexterr.htm
<EXIT>
<endif>
```

MRE Callable Exits

The following table lists the user written exits that are part of the MRCEXT module. Except for MRCEXT3, these exits are for the MRE Repository:

Exit	Description
MRCEXT0	<p>Authenticates security credentials and sets the preference file and domains the user can access. This exit applies to MRE Java users only and must be used in conjunction with MRCEXT1. Administrators must use a different logon page, so as not to invoke this exit.</p> <p>Note:</p> <ul style="list-style-type: none"> Information Builders recommends using the MRCEXT5 exit instead of MRCEXT0. Enabling MRCEXT1 automatically enables MRCEXT0, provided that MRCEXT5 is not enabled.
MRCEXT1	Overrides default MRE WebFOCUS Script File (WFS) variable values. This exit is required for MRCEXT0 and MRCEXT5.
MRCEXT2	Determines which users have access to Shared Reports. MRCEXT2 is automatically invoked if it is coded and MRCEXT5 is active.
MRCEXT3	Uses an alternate algorithm for encrypting and decrypting WebFOCUS cookies, replacing the WebFOCUS internal cookie algorithm.

Exit	Description
MRCEXT4	Authenticates the MRE user ID and password only. This exit allows the user ID and password to be authenticated outside of MRE. However, the user's privileges and IDs are still managed from within the MRE Administrator tool.
MRCEXT5	Replaces the MRCEXT0 exit and the MRE User Administrator tool. This exit authenticates security credentials, sets the user's preference file, assigns the user type (for example, MRE Administrator or Domain Admin), and determines which domains the user can access.

MRCEXT Header File for MRE Callable Exits

This C header file defines the various exits used by MRE. The Web administrator may supply any that are desired. On Microsoft systems, these exits must be in a DLL called `mrcext.dll` (`libmrcext.so` or `libmrcext.sl` on UNIX). If the exits are all in a C file called `mrcext.c`, this DLL may be created with the command `CL /GD /LD mrcext.c`. The DLL will then have to be installed with the rest of the MRE CGI.

```

/*-----
* Copyright (c) 1998 Information Builders, Inc. All rights reserved.
*
* _Name_      ==> MRCEXT   H          _Opsys_    ==> CMS
* _Release_   ==> 72      _Product_   ==> FOC
* _Description_ ==>
*
* _Notes_     ==>
*
* _History_:
* Date Time Who Proj      Project Title
* =====
* 000409 1319 WJF 36239 USER EXIT FUNCTION MREXT2 NEEDS TO BE ENHANCED
* 990220 1508 WJF 30293 WEBFOCUS: REMOVE STATICS FROM CGI
* 990208 1555 WJF 30289 WEBFOCUS: MRE RETURNS OF 02/08/99
* 990204 2012 WJF 30213 WEBFOCUS: TRACE LEVELS IN CGI
* 981130 1601 WJF 29232 WEBFOCUS: MRE CHANGES NOV 30.
* 981019 1605 PXX 28648 MR CGI UPDATES HTML USERS SHARED
* 980302 1311 KWL 26289 NSAPI VERSION OF WEBFOCUS CGI
* 980116 1026 LAM 25980 MR CGI UPDATES JAN 14
*
* END %&%
*-----*/

```

```
#ifndef MRCEXT_H
#define MRCEXT_H

#ifdef __cplusplus
extern "C" {
#endif

#ifdef _WIN32
#define DLLENTRY __declspec(dllexport)
#else
#define DLLENTRY
#endif

#define MR_EXT_SUCCESS 2000L          /* Successful return code from Exits
*/
/* This exit will be invoked to do UserID/PW validation, and to supply*/
/* the Domain and UserFile to be associated with this user*/

long DLLENTRY mrcext0 (
    char *user          /* Input: Signon UserID*/
    ,char *pass         /* Input: and Password*/
    ,char *userFile     /* Output:The "page" for UserID*/
    ,char *domains[]   /* Input: Table of known Domains*/
    ,long count        /* Input:  and how many of them*/
    ,char **usedomain  /* Output: The Domain for UserID*/
    ,char *trace       /* Output: Area to assign value*/
    ,long traceMaxLen  /* Input:  Size of output area*/
);

/* This exit will be invoked to supply values to some internal variables*/
/* (on a per user basis, if desired)*/

long DLLENTRY mrcext1 (
```

```

        char *user          /* Input: Signon UserID*/
        ,char *pass        /* Input: and PassWord*/
        ,char *name        /* Input: Variable to replace*/
        ,char *value       /* Output: Area to assign value*/
        ,long valueMaxLen  /* Input: Size of output area*/
        ,char *trace       /* Output: Area to assign value*/
        ,long traceMaxLen  /* Input: Size of output area*/
    );

struct mrcect2_struct
{
    char *user;              /* full user name */
    char *userid;           /* userid for this user */
    char *userFile;         /* filename for this user */
    struct mrcect2_struct *next; /* pointer to next entry */
};

long DLENTY mrcect2 (char *domain /* Input: The Domain */
                    ,struct mrcect2_struct **table /* pointer to start of
table */
                    ,long flag /* 1 = fill in table 2= free
table*/
                    ,char *user /* Input: Signon UserID */
                    );

long DLENTY mrcect3 (char *input /* Input : Input data */
                    ,long input_length /* Input : Input length */
                    ,char **output /* Output: output data */
                    ,long EnDeFrflag /* Input : 1 encrypt
** 2 decrypt
** 3 free output buffer
*/
                    );

/* return 0 for pass checked */
long DLENTY mrcect4 (char *uid /* user id */
                    ,char *pass /* pass to check */
                    );

long DLENTY mrcect5 (
    char *user /* Input: Signon UserID*/
    ,char *pass /* Input: and PassWord*/
    ,char *userFile /* Output: The "page" for UserID*/
    ,char *domains[] /* Input: Table of known
Domains*/
    ,long count /* Input: and how many of them*/
    ,char **usedomain /* Output: The Domain for UserID*/
    ,char *trace /* Output: Area to assign value*/
    ,long traceMaxLen /* Input: Size of output area*/
    ,char *usertype /* Output: user type */
);

#ifdef __cplusplus
}
#endif

```

MRCEXT0 Exit

Tip:

Since the MRCEXT5 exit applies to all user types, it is recommended that you use MRCEXT5 instead of MRCEXT0. With MRCEXT5, a single logon page may be used for any type of user. For more information, see *MRCEXT5 Exit* on page 4-32.

The MRCEXT0 exit allows developers to create a custom authentication program for use within the MRE. This means that the MRE User Administrator tool is not used since all user administration is handled externally. The MRCEXT0 exit applies only to MRE Java users and must be used in conjunction with MRCEXT1.

When using the MRCEXT0 exit, the user ID and password are passed to the CGI using the MRSIGNON action; the CGI will then call the external program to authenticate the user ID. Inbound to this call are the user ID and password and a set of all domains within the MRE Repository. The program returns with the user's domain, a unique user-based index file, and a return code. An MR_EXT_SUCCESS return code of 2000 is successful. All other return codes should be between 3000-9999 and will be set using the exit. Users will only have access to the domain passed back from the program.

Users created and authenticated using this external method will have a physical directory created in the MRE Repository structure for each user. The user ID will be the directory name, and all MyReports created by the user will be stored in this directory. An index of entries in this directory is stored inside the user-based index file. This page acts as the user's index of reports.

Syntax

How to Create the MRCEXT0 Exit

```
long mrcext0 (char user, char pass, char userfile,  
             char domains, long count, char usedomain,  
             char trace, long trace_max_len)
```

where:

user

Is the signon user ID.

pass

Is the signon password.

userfile

Is the user preference file.

domains

Is the table of known domain HREFs.

count

Is the number of domain HREFs in the table.

usedomain

Is an array of pointers to the domains the user may access. The array is allocated as count + 1. The entry after the last valid entry should be set to NULL.

trace

Is the trace area to assign the value.

trace_max_len

Is the maximum size of the trace output line.

Example **Creating the MRCEXT0 Exit**

Based on the results returned from the security repository (possibly LDAP or another repository), this exit sets the Java user type and the domains that can be accessed by MRE.

Note: A C header file is required to compile the exit. For a sample C header file, see *MRCEXT Header File for MRE Callable Exits* on page 4-17.

```

/*
*****
* MRE exit 0:
*****
*/
long mrcext0 (char *user, char *pass, char *userfile,
             char **domains, long count, char **usedomain,
             char *trace, long trace_max_len)
{
    int      i,j,k;
    PARMS    parms;

    /* Authenticate user */
    if( (getSecurCred(user,pass,&parms)) !=0)
        return(MR_EXT_FAIL);

    /* set mre user file */
    sprintf(userfile,"%s.htm",user);

    /* set domains to use */
    for(i=0,k=0;i<parms.nDmns && i<count;i++)
    {
        for(j=0;j<count;j++)
        {
            /* compare domain HREF's */
            if( (strcmp(*(domains+j),parms.szDmns[i])) ==0)
            {
                *(usedomain+k) = *(domains+j);
                k++;
            }
        }
        *(usedomain+k)=NULL;
    }

    /* When tracing is enabled, write to WebFOCUS trace file */
    sprintf(trace,"mrcext0:user=%s userfile=%s (domains=%ld
use=%d)",user,userfile,count,k);

    return MR_EXT_SUCCESS;
}

```

MRCEXT1 Exit

The MRCEXT1 exit is available for use within the MRCEXT0 or MRCEXT5 exits. It can be used to dynamically change the runtime environment variables. These variables include the following:

- **MR_BASE_DIR** is the physical location of the repository root directory.
- **MR_BASE_FILE** is the repository file name.
- **MR_BASE_HTML** is the directory where published launch pages are written.
- **IBI_DOCUMENT_ROOT** is the location where the Java applets receive accessory files.
- **MR_TEMPLATE_DIR** is the location of template files for the WebFOCUS Publish utility.

The MRCEXT0/MRCEXT5 and MRCEXT1 exits must reside in the MRCEXT module in the same directory as the WebFOCUS CGI. Both of these exits are called each time. A return code of 2000 must be returned for both MRCEXT0/MRCEXT5 and MRCEXT1.

Reference

Setting Variables and HTML Form Actions for MRCEXT1

To enable the use of the MRCEXT1 exit, the following variable must be set in the site.wfs file:

```
MR_USER_ADMIN=MR_USER_ADMIN_MODE1
```

The MRCEXT0/MRCEXT5 and MRCEXT1 exits also require the following code to be set in the site.wfs file:

```
<IF> IBIMR_action EQ "MR_SIGNON" OR IBIMR_action EQ "MR_CHANGE_PASS"
IBIMR_adminmode=MR_USER_ADMIN_MODE1
<endif>
```

Optionally, if you are not using ReportCaster, you can add the following HTML form action to site.wfs instead of the IBIMR_action specified in the code above:

```
<INPUT TYPE="HIDDEN" NAME="IBIMR_adminmode" VALUE="MR_USER_ADMIN_MODE1">
```

For more information about MRE signon forms, see *MRCEXT5 Exit* on page 4-32. For more information about variables, see *WebFOCUS Script (WFS) Commands* in Chapter 5, *WebFOCUS Administration*.

Syntax

How to Create the MRCEXT1 Exit

```
long mrcext1 (char user, char pass, char name, char value, long max,  
             char trace, long trace_max_len)
```

where:

user

Is the signon user ID.

pass

Is the signon password.

name

Is the WFS variable name. Possible values are:

MR_BASE_DIR is the physical location of the repository root directory.

MR_BASE_FILE is the repository file name.

MR_BASE_HTML is the directory where published launch pages are written.

IBI_DOCUMENT_ROOT is the location where the Java applets receive accessory files.

MR_TEMPLATE_DIR is the location of template files for the WebFOCUS Publish utility.

value

Is the area to assign a new value.

max

Is the size of the output area.

trace

Is the area to assign the value.

trace_max_len

Is the size of the output area.

Example **Creating the MRCEXT1 Exit**

This sample returns a success code of 2000. It does not change any of the default values.

Note: A C header file is required to compile this exit. For a sample C header file, see *MRCEXT Header File for MRE Callable Exits* on page 4-17.

```

/*
*****
* MRE exit 1:
*****
*/
long mrcext1 (char *user, char *pass, char *name, char *value, long max,
             char *trace, long trace_max_len)
{
    /* example to override MR_BASE_DIR value -- otherwise use defaults
     * if (strcmp(name,"MR_BASE_DIR") == 0)
     *     strcpy(value,"c:\ibi\WebFOCUS436\...");
     */
    return(MR_EXT_SUCCESS);
}

```

MRCEXT2 Exit

The MRCEXT2 exit determines which users have access to Shared Reports. MRCEXT2 is automatically invoked if it is coded and MRCEXT5 is active. An MR_EXT_SUCCESS return code of 2000 is successful. All other values indicate failure.

Syntax

How to Create the MRCEXT2 Exit

```
long mrcext2 (char *info, struct mrcext2_struct **table, long flag, char *user, ext_trc *ibitrc)
```

where:

info

Is a value based on the settings in *flag*:

- If the *flag* variable is set to 1, *info* contains the domain name.
- If the *flag* variable is set to 2, *info* contains nothing.
- If the *flag* variable is set to 3, *info* contains the user's storage page (userfile).
- If the *flag* variable is set to 4, *info* contains the user ID.

table

Is a table of Shared Reports users that is based on the settings in *flag*:

- If the *flag* variable is set to 1, *table* populates the list of users (linklist) with the Shared Reports usertype.
- If the *flag* variable is set to 2, *table* deletes the list of users (linklist) with the Shared Reports usertype.
- If the *flag* variable is set to 3 or 4, *table* contains the matching user entry.

flag

Is the action to be taken by the exit. This is called by MRE when needed.

- 1 (GET_USER_TABLE): Load table with all Shared Reports users for the domain contained in the *info* parameter. This action is invoked when the Shared tab is selected.
- 2 (DELETE_USER_TABLE): Delete table.
- 3 (FIND_USER_HTM): Look up storage page. This action is invoked when selecting a shared user. It populates the table with that user's entry.
- 4 (FIND_USER_ID): Look up user ID. This action is invoked prior to running a report request. It populates the table with that user's entry.

user (input)

User ID that logs onto MRE.

ibitrc (output)

Writes to standard WebFOCUS trace files.

Example Creating the MRCEXT2 Exit

This sample loads the users who have access to Shared Reports for a domain.

Note: A C header file is required to compile this exit. For a sample C header file, see *MRCEXT Header File for MRE Callable Exits* on page 4-17.

```

long mrcext2 (char *info,struct mrcext2_struct **table,long flag,char
*user,
            ext_trc *ibitrc)
{
    FILE    *fp;
    int     i;
    long    lRc;
    TABLE  utable;
    struct mrcext2_struct *users, *last, *temp;

    switch (flag)
    {
        /* 1> create linklist table of users */
        case GET_USER_TABLE:
            last=NULL;
            *table=NULL;

            if ((fp=fiOpen(szTrace)) == NULL)
                return(MR_EXT_FAIL);

            /* load users in domain with a usertype of shared      */
            /* MRE will filter for only users with a shared usertype */
            /* For performance reasons you may choose to filter the */
            /* users prior to MRE's filtering                        */
            while(!(fiFetch(fp,&utable,szTrace))
            {
                for(i=0;i<utable.nDmns;i++)
                {
                    if( (strcmp(info,utable.szDmns[i])) ==0)
                    {
                        users = malloc(sizeof(struct mrcext2_struct));
                        if(!last)
                            *table = users;
                        else
                            last->next = users;
                        last = users;
                        strcpy(users->user,utable.szUserDesc);
                        strcpy(users->userid,utable.szUserId);
                        strcpy(users->userFile,utable.szUserFile);
                        strcpy(users->flags,utable.szAuth);
                        users->next = NULL;
                        break;
                    }
                }
            }
            fiClose(fp);
            /* set return code */
            if(*table)
                lRc=MR_EXT_SUCCESS;
            else
                lRc=MR_EXT_FAIL;
            break;
    }
}

```

```
/* 2> free linklist table entries */
case DELETE_USER_TABLE:
    users=*table;
    while(users)
    {
        temp=users->next;
        free(users);
        users=temp;
    }

    lRc=MR_EXT_SUCCESS;
    break;

/* 3> lookup user using htm file */
case FIND_USER_HTM:
if(!srch(&utable,info,utable.szUserFile,ibitrc))
    {
        strcpy((*table)->user,utable.szUserDesc);
        strcpy((*table)->userid,utable.szUserId);
        strcpy((*table)->userFile,utable.szUserFile);
        strcpy((*table)->flags,utable.szAuth);
        lRc=MR_EXT_SUCCESS;
    }
    else
        lRc=MR_EXT_FAIL;
    break;

/* 4> lookup user using userid */
case FIND_USER_ID:
    if(!srch(&utable,info,utable.szUserId,ibitrc))
    {
        strcpy((*table)->user,utable.szUserDesc);
        strcpy((*table)->userid,utable.szUserId);
        strcpy((*table)->userFile,utable.szUserFile);
        strcpy((*table)->flags,utable.szAuth);
        lRc=MR_EXT_SUCCESS;
    }
    else
        lRc=MR_EXT_FAIL;
    break;
/* Unknown flag */
default:
    lRc=MR_EXT_FAIL;
    break;
}

if(lRc==MR_EXT_SUCCESS)
    TRACE("OK");
else
    TRACE("Failed");

return(lRc);
}
```

MRCEXT3 Exit

The MRCEXT3 exit allows developers to change the encryption algorithm that is used to encrypt MRE and WebFOCUS cookies.

To enable the MRCEXT3 exit, you must set position 5 of MR_PGM_FLAGS to Y. An MR_EXT_SUCCESS return code of 2000 is successful. All other values indicate failure. For more information about MR_PGM_FLAGS, see Chapter 5, *WebFOCUS Administration*.

Syntax

How to Create the MRCEXT3 Exit

```
long mrcext3 (char input, long input_length, char output,  
             long EnDeFrflag)
```

where:

input

Is the input cookie data.

input_length

Is the input length.

output

Is the output cookie data.

EnDeFrflag

Is the action flag. Possible values are:

- 1 to encrypt.
- 2 to decrypt.
- 3 to free output buffer.

Example **Creating the MRCEXT3 Exit**

This sample reverses the information to be stored in the WebFOCUS and MRE cookies.

Note: A C header file is required to compile this exit. For a sample C header file, see *MRCEXT Header File for MRE Callable Exits* on page 4-17.

```
/*
*****
* MRE exit 3:
*****
*/
long mrcext3 (char *input, long input_length, char **output, long
EnDeFrflag)
{
    switch (EnDeFrflag)
    {
        case 1: /* encrypt */
            *output = _strrev( _strdup( input ) ) ;
            break;
        case 2: /* decrypt */
            *output = _strrev( _strdup( input ) ) ;
            break;
        case 3: /* free memory */
            if(input)
                free(input);
            break;
    }
    return MR_EXT_SUCCESS;
}
```

MRCEXT4 Exit

The MRCEXT4 exit allows a developer to create a program that authenticates the user ID and password outside of MRE using an external program. This program can be coded to check against an external security subsystem. The user IDs and privileges must be created in the MRE Repository and managed with the User Administrator tool. All domain privileges and user rights will be maintained within the MRE Repository, unlike the MRCEXT0 and MRCEXT5 exits, which handle all user management externally.

To enable the MRCEXT4 exit, you must set position 7 of MR_PGM_FLAGS to Y. A return code of 0 indicates success. All other values indicate failure. For more information about MR_PGM_FLAGS, see Chapter 5, *WebFOCUS Administration*.

Note: This exit is not needed when using the MRCEXT0 or MRCEXT5 exits.

Syntax**How to Create the MRCEXT4 Exit**

```
long mrcext4 (char uid, char pass)
```

where:

uid

Is the user ID.

pass

Is the password.

Example**Creating the MRCEXT4 Exit**

This sample checks LDAP to authenticate the user ID and password.

Note: A C header file is required to compile this exit. For a sample C header file, see *MRCEXT Header File for MRE Callable Exits* on page 4-17.

```
/*
*****
* MRE exit 4:
*****
*/
long mrcext4 (char *uid ,char *pass)
{
    char uid_filter[100]="uid=";
    int results = 1;

    strcat (uid_filter,uid) ;

    results=authenticate (uid_filter,pass) ;
    return(results) ;
}

int authenticate(char *uid_filter, char *passwd)
{
    LDAP *ld;
    LDAPMessage *result,
    *e;
    char *dn;
    int value =1;
    int rc;
    int iEntries;
    char *MY_FILTER = uid_filter;
```

```

        if ((ld=ldap_init(MY_HOST,MY_PORT))==NULL)
return ldap_get_errno(ld);

        /* locate uid */
        rc=ldap_search_s(ld,MY_SEARCHBASE, LDAP_SCOPE_SUBTREE, MY_FILTER, NULL,
0 ,&result);
        if(rc!= LDAP_SUCCESS)
return ldap_get_errno(ld);

        iEntries=ldap_count_entries(ld,result);
        if(iEntries==-1)
return ldap_get_errno(ld);

        if(e=ldap_first_entry(ld,result))
        {
            if((dn=ldap_get_dn(ld,e)) != NULL)
            {
                if(strlen(dn)==0 || (strlen(passwd)==0))
                    return(1);

                /* bind user with password and dn */
                rc=ldap_simple_bind_s(ld,dn,passwd);
                if ( rc != LDAP_SUCCESS )
                    return ldap_get_errno(ld);
                value=0;
                ldap_memfree(dn);
            }
        }

        ldap_msgfree(result);
        ldap_unbind(ld);
        return(value);
    }

```

MRCEXT5 Exit

The MRCEXT5 exit provides the same functionality as the MRCEXT0 exit, plus additional functionality. The additional functionality is that MRCEXT5 allows for the user's privileges to be passed back, and it can be called independent of MRCEXT1.

To enable the MRCEXT5 exit, the MRCEXT1 exit must be enabled and position 8 of MR_PGM_FLAGS must be set to Y. An MR_EXT_SUCCESS return code of 2000 is successful. All other values indicate failure. For more information about MR_PGM_FLAGS, see Chapter 5, *WebFOCUS Administration*.

Note: If you are using MRCEXT5, you must rewrite your exit program to include calls to MRCEXT2 in order to support the Shared Reports feature of Managed Reporting, and the scheduling of My Reports with ReportCaster. The User Administrator tool in Domain Builder is not supported by MRCEXT5.

Syntax

How to Create the MRCEXT5 Exit

```
long mrcext5 (char user, char pass, char userfile,  
             char domains, long count, char usedomain,  
             char trace, long trace_max_len, char usertype)
```

where:

user

Is the signon user ID.

pass

Is the signon password.

userfile

Is the storage page for the user ID.

domains

Is the table of known domain HREF's.

count

Is the number of domain HREF's in the table.

usedomain

Is an array of pointers to the domains the user may access. The array is allocated as count + 1. The entry after the last valid entry should be set to NULL. This value is ignored when the user type is set to admin.

trace

Is the buffer to assign the trace value.

trace_max_len

Is the maximum buffer size of the output trace area.

usertype

Is the user's privilege type. For multiple user privilege types, delimit the user types with a comma (for example, active,robot). Possible values are:

active indicates the user account is enabled.

inactive indicates the user account is disabled.

admin indicates the user has MRE Administrator privileges.

dadomains=untitled%2csalesdom indicates this Domain Admin is restricted to administering only the default domain, which has the internal references "untitled" and "Sales Domain." The characters "%2c" delimit the domains when there is more than one domain.

dataserver={true|false} indicates whether or not the Domain Admin has access to the Data Server feature.

domadmin indicates the user is a Domain Admin.

htmluser indicates the user is a HTML user.

runonly indicates the Java user cannot save reports.

The following user privilege types are appended to an already specified usertype (for example, active,robot). Possible values are:

shared indicates the user can share reports.

robot indicates the user has scheduling privileges.

emr indicates the desktop user has the Financial Modeling Language (FML) feature enabled.

sync indicates the user has PDA Sync privileges.

Example

Creating the MRCEXT5 Exit

Based on the results returned from the security repository (possibly LDAP or another repository), this exit sets the user type access rights and the domains that can be accessed by MRE.

Note: A C header file is required to compile this exit. For a sample C header file, see *MRCEXT Header File for MRE Callable Exits* on page 4-17.

```

/*
*****
* Copyright (c) 2001 Information Builders, Inc. All rights reserved.
*
* Program ID: mrcext.c
* Module ID : WinNT= mrcext.dll
*             UNIX = libmrcext.so or libmrcext.sl
*
* This code is sample code, provided as-is.
* No warranties are made as to its correctness or
* suitability for any purpose.
*
* Description: Sample X-PLATFORM WebFOCUS MRE User exit (MREXT5).
*
*****
*/
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <string.h>
#include <errno.h>
#include <malloc.h>
#include "mrcext.h"
#define SECFILE "mrcext.dat"
#define SECMAXSZ 1024
#define MAXUSEDM 20
#define MR_EXT_FAIL 3001L
typedef struct tagPARMS /* temp table storage */
{
    char          szAuth[48];
    char          szDmns[MAXUSEDM][48];
    int           nDmns;
} PARMS;

long getSecurCred(char *, char *,PARMS *);
/*
*****
* MRE exit 1:
*****
*
* Description:
* Overrides default MRE WFS Variable values (required for exits 0 and 5).
* Does not change any of the default values.
*

```

```

*****
*/
long mrcext1 (char *user, char *pass, char *name, char *value, long max,
             char *trace, long trace_max_len)
{
    /* example to override MR_BASE_DIR value -- otherwise use defaults
     * if (strcmp(name,"MR_BASE_DIR") == 0)
     *     strcpy(value,"c:\ibi\WebFOCUS436\...");
     */
    return(MR_EXT_SUCCESS);
}

/*
*****
* MRE exit 5:
*****
long mrcext5 (char *user, char *pass, char *userfile,
             char **domains, long count, char **usedomain,
             char *trace, long trace_max_len, char *usertype)
{
    int      i,j,k;
    PARMS    parms;
    /* init usedomains */

    *usedomain=*domains;
    *(usedomain+count)=NULL;

    /* Authenticate user and populate parms table */
    if( (getSecurCred(user,pass,&parms))!=0)
        return(MR_EXT_FAIL);

    /* set users access rights */
    strcpy(usertype,parms.szAuth);

    /* set mre user file and domains to use */
    sprintf(userfile,"%s.htm",user);

    /* set domains to use */
    for(i=0,k=0;i<parms.nDmns && i<count;i++)
    {
        for(j=0;j<count;j++)
        {
            /* compare domain HREF's */
            if( (strcmp(*(domains+j),parms.szDmns[i])) ==0)
            {
                *(usedomain+k) = *(domains+j);
                k++;
            }
        }
    }
}

```

```

/* When tracing is enabled, write to WebFOCUS trace file */
    sprintf(trace,"mrcext5:user=%s userfile=%s usertype=%s (domains=%ld
use=%d)",
            user,userfile,usertype,count,k);

    if(!k) /* no domains to use */
        return(MR_EXT_FAIL);

    *(usedomain+k)=NULL;

    return MR_EXT_SUCCESS;
}

```

Compiling MRE Callable Exits

The following is a sample make file for compiling an MRE callable exit on a Sun Microsystems® platform:

```

SHELL = /bin/ksh

DEFINES=-D_REENTRANT -DFT_UNIX
CFLAGS=-g ${DEFINES} -I${EDAHOME}/etc
LIBS=-G -lc

all: mrcext

clean:
    rm -f mrcext.o

mrcext: mrcext.o
    cc -o libmrcext.so mrcext.o ${LIBS}

```

The following is a sample for compiling an MRE callable exit on an IBM OpenEdition® UNIX Systems Services (OS/390®) platform:

```
cc -o libmrcext.so -W c,dll,expo -W l,dll mrcext.c
```

Note: For more information about make files, see your operating system documentation.

WebFOCUS DBA Exits

WebFOCUS DBA exits enable Master Files to use replaceable parameters that are retrieved using this exit. This exit is usually used to allow DBA `RESTRICT=VALUE`, where `VALUE` is the statement (or statements) to be formed using external logic.

The WebFOCUS DBA Exit requires a custom C program compiled into a module called XDBSC. This file must be stored in the directory pointed to by the IBICPG variable defined in the WebFOCUS Reporting Server environment. A DBA Exit will be called when the server-based Master File is accessed. The symbolic variable used for the call of this exit is an @ character. The ID that is passed to the exit is the connected user ID.

Reference

Sample Master File

In the following sample Master File, the symbolic variables (for example, @FIELD) can be placed anywhere in the VALUE statement, so complex logic can be implemented. A VALUE statement can be a single parameter in which the exit provides the entire restriction statement.

```
/* END */
/* DBA=DOGBERT, $ */
/* USER=SSFULL, ACCESS=R, $ */
/* USER=JILL, ACCESS=R, RESTRICT=VALUE, NAME=SYSTEM, */
/* VALUE = @FIELD @EQUALITY '@VALUE', $ */
/* USER=ROBERT, ACCESS=R, RESTRICT=VALUE, NAME=SYSTEM, */
/* VALUE = COUNTRY EQ '@ROBERT', $ */
/* USER=NATALIE, ACCESS=R, RESTRICT=VALUE, NAME=SYSTEM, */
/* VALUE = COUNTRY EQ 'JAPAN', $ */
/* USER=TERRY, ACCESS=R, RESTRICT=VALUE, NAME=SYSTEM, */
/* VALUE = COUNTRY EQ '@VARIOUS', $ */
```

Syntax

How to Call Parameters for the WebFOCUS DBA Exit

```
void xdbsc (long *iFunc, char *user_ID, char *secGrp, char *table,  
char *msg, void *token, long *rc, char *access, char *symbol, char *value)
```

where:

iFunc

Is input to the WebFOCUS DBA Exit. This value defines the type of exit to call. Possible values are:

- 1 - XDBSCP (POST) is called to determine what security group the user should be in for this table.
- 2 - XDBSCF (FETCH) is called for each symbolic variable specified in the value restriction expressions for the security group resulting from XDBSCP.

user_ID

Is the user's signon user ID.

secGrp

For XDBSCP, there is input to and output from the exit. On input, this is the security group that the user signed on with or has been given by the logon exit. Upon exiting, this is the security group that should be used for this query. For XDBSCF, input to the exit is from the previous XDBSCP call.

table

Is the input parameter that indicates which table the query is against. This is a zero-delimited string with a maximum length of 20, including the string termination character (\0).

msg

Is the output parameter that displays an error message only in the case of a non-zero return code (indicating XDBSCF failure).

token

Is input to the WebFOCUS DBA Exit.

rc

Is the return code. A return code of 0 indicates success. A return code of 1 indicates failure and aborts the query.

access

Is only applicable to XDBSCF exits, and should be passed as a dummy input parameter for XDBSCP exit calls.

symbol

Is only applicable to XDBSCF exits, and should be passed as a dummy input parameter for XDBSCP exit calls. This is the symbol (parm) name without the '@' prefix. This is a zero-delimited string with maximum length of 68, including the string termination character (\0).

value

Is only applicable to XDBSCF exits, and should be passed as a dummy output parameter for XDBSCP exit calls. This parameter indicates the value to be substituted for the symbol. If multiple values are specified, they must be separated by \0 and \0\0 to indicate the end of list. This parameter has a maximum length of 4096, including the string termination character (\0).

Example Using the WebFOCUS DBA Exit

The following example uses the *Sample Master File* on page 4-38 to limit user access to data sources.

Compilation and usage of this exit follows the standard methods for user subroutines, which is gencpgm on most platforms. Next, copy the exit to the user directory of EDACONF, or set the environment variable IBICPG to the name of the actual working directory. For more information, see the *iWay Stored Procedures Reference* manual.

```
/* XDBSC DBA Security Exit                                     */
/*                                                         */
#include <string.h>
#include <stdio.h>
#include <errno.h>

#define F_POST 1 /* XDBSCP */
#define F_FETCH 2 /* XDBSCF */

void xdbsc(long *iFunc, char *user_ID, char *secGrp, char *table,
           char *msg, void *token, long *pRC,
           char *access, char *symbol, char *value)
/* Last 3 args for F_FETCH only */
{
    char ubuf[9], gbuf[9];
    FILE *logFile;

    memcpy(ubuf, user_ID, 8);
    ubuf[8] = '\0';
    memcpy(gbuf, secGrp, 8);
    gbuf[8] = '\0';
    errno = 0;
    /*
    /* The logging intended in this example is to confirm
    /* behavior in "edastart -t" mode and then turn
    /* off logging or comment it out until needed again.
    /*
    */
    */
    */
    */
}
```

```

/* This example is coded with a compilation flag for IBM Mainframe */
/* to activate. It writes to stderr as the simplest way to capture */
/* output. */
/* If you are in an IBM Mainframe environemnt, make sure that you */
/* compile with a DEFINE(IBMFMF=1) and have STDOUT declared in the */
/* start up JCL. In this environment, stderr is routed to stdout */
/* for "edastart -t" mode and a trace file for server mode use. */
/* */
/* On non IBM Mainframe environments the default is to code as a */
/* local file for "edastart -t" mode. For a server, use a full path */
/* so that the log is common to all agents and remains between */
/* agents uses. For example: */
/*   logFile = fopen("/home/iadmin/ibi/srv51/ffs/xdbsc.log", "a"); */
/*   logFile = fopen("EDA:[IADMIN.IBI.SRV51.FFS]XDBSC.LOG", "a"); */
/*   logFile = fopen("c:\tmp\xdbsc.log", "a"); */
/* */
/* Note: Only IBM Mainfram needs the IBMFMF compilation flag. */
/* */
#ifdef IBMFMF
    logFile = stderr;
#else
    logFile = fopen("xdbsc.log", "a");
    if (errno != 0)
        printf("!!!! errno = %d: %s\n", errno, strerror(errno));
#endif
    strcpy(msg, " "); /* Clear buffer for fprintf */

    if (logFile)
    {
        if (*iFunc == F_POST)
        { /* Change/Add/Format values needed for site */
            fprintf(logFile,
                "xdbsc: iFunc=%ld, ID=%s, grp=%s, table=%s, msg=%s, token=%p,
rc=%ld\n",
                *iFunc, ubuf, gbuf, table, msg, token, *pRC);
#ifdef IBMFMF
            fclose(logFile);
#endif
        }
    }

    *pRC = 0; /* Successful return */

    if (*iFunc == F_POST)
        return;

/*
/* Place your site specific if else logic here, using the */
/* passed in values as criteria. Nothing above this section */
/* should have major changes except for any log control you */
/* may want performed. Primary work is done in this section.*/
/* */
if (
    (memcmp(user_ID, "edaqa0 ", 8) == 0) ||
    (memcmp(user_ID, "EDAQA0 ", 8) == 0)
)
{

```

```

if (strcmp(symbol, "FIELD") == 0)
strcpy(value, "COUNTRY");
else if (strcmp(symbol, "EQUALITY") == 0)
strcpy(value, "EQ");
else if (strcmp(symbol, "VALUE") == 0)
strcpy(value, "FRANCE");
else if (strcmp(symbol, "ROBERT") == 0)
strcpy(value, "ENGLAND");
else {
    strcpy(value, "DBA Failure");
    strcpy(msg, "XDBCS DBA Exit: If Else Logic Failure!\n");
    *pRC = -1; /* 0 causes the value to still substitute (so value */
              /* should be one that is unlikely to match a */
              /* real value so it always returns no records) */
              /* and gives the standard ACCESS LIMITED BY */
              /* PASSWORD message. */
              /*
              /* Non Zero causes abort query with msg value and */
              /* (FOC047) THE USER DOES NOT HAVE SUFFICIENT */
              /* ACCESS RIGHTS TO THE FILE: *table */
              /* and BYPASSING TO END OF COMMAND message. */
              /*
              /* So depending on need you might use a value of */
              /* zero for a fail. */
}
}
else if (memcmp(user_ID, "NORMAN ", 8) == 0)
{
    strcpy(value, "JAPAN");
}
else
{
    strcpy(value, "ITALY");
}

if (logFile)
{
if (*iFunc == F_FETCH)
{ /* Change/Add/Format values needed for site */
fprintf(logFile,
"xdbsc: iFunc=%ld, ID=%s, grp=%s, table=%s, msg=%s, token=%p,
rc=%ld",
        *iFunc, ubuf, gbuf, table, msg, token, *pRC);
fprintf(logFile, " access=%s, symbol=%s, value=%s\n",
        access, symbol, value);
#ifdef IBMMF
fclose(logFile);
#endif
}
}
}
}

```

Pre-Verify User ID Exit

The Pre-Verify User ID Exit (PVUIDXT) may be used with Already Verified Processing to customize a trusted node security scheme. This exit matches a user ID with the connecting partner nodes identified by the following:

Communications Protocol	Partner Communications Identifier
TCP/IP	The network ID (an alphanumeric representation of the IP address).

This exit allows or disallows connection to a WebFOCUS Reporting Server and may replace the input user ID with an appropriate user ID for the host security subsystem. It may also provide a password to be included in the verification procedure.

For example, a Windows client accesses OS/390 data through a UNIX Hub server. With this configuration, the user ID and password can be verified on the UNIX Hub server, and a secure connection can be passed on to OS/390. This configuration uses Already Verified Processing, providing trusted node security.

The remainder of these topics provides an overview of how a configuration can be set up with a Hub server installed as a trusted node. If this exit is not installed, no connections are rejected, based on the partner communications ID.

Syntax

How to Use PVUIDXT Calling Sequences

In order for further validation of a user ID and a router address, the exit is called with the following sequence:

```
/* pvuidxt( pPartner_Logical_Name, pPartner_Communications_ID, */
/*          pUserid, pPassword, pRC ) ; */
/*                                          */
void pvuidxt( char *, char *, char *, char *, long * ) ;
```

where:

pPartner_Logical_Name

Points to the string containing the connecting partnerentity_name.

pPartner_Communications_ID

Points to the string containing the communication ID for TCP/IP (the IP address). This is a variable length alphanumeric string. This value is not supplied for SNA connections.

pUserid

Points to an area containing the user ID provided by the client application. It is passed in uppercase. If the user exit replaces the incoming user ID, it must not use a length greater than or equal to the input string length.

pPassword

Points to an area, 8 bytes in length, into which the user exit can place a password associated with the incoming user ID from the specified partner node. If this password field contains a password to be used at the time of verification, pRC must be set equal to 1 (TRUSTED_EXIT_SUCCESS_USE_PSW).

pRC

Is the return code from the exit. Possible values are:

- 0 indicates the user ID/entity name combination has been validated and the user ID will be checked against your system security package.
- 1 indicates the user ID and password will be verified.
- 1 indicates the validation check has failed and the user will be disconnected.

Syntax

How to Apply Long Sequences

The `trustos` routine is called before `PVUIDXT` and determines whether the client operating system security is accepted by the server security system.

```
long trustos( unsigned char ) ;
```

where:

`trustos`

Returns a value of zero or one. The default returned by OS/390, VM, and UNIX is one; all other platforms or operating systems return zero. A user who wants a different result will need to write a `trustos` routine.

0 indicates that the client operating system security is unacceptable.

1 indicates that the client operating system security is acceptable.

`unsigned char`

Is the already verified code for client applications.

0 indicates that the No already verified code is available.

1 indicates that the OS/390 client is already verified.

2 indicates that the OpenVMS client is already verified.

3 indicates that the UNIX client is already verified.

4 indicates that the VM client is already verified.

5 indicates that the Windows NT/2000 client is already verified.

Note: Your system administration staff can create their own conventions for other known security packages.

For UNIX and Windows NT/2000, a sample `PVUIDXT` logon is located in `$EDAHOME/etc`.

For OS/390, a template for the `PVUIDXT` logon is located in `qualif.EDACTL.DATA(PVUIDXTA)`.

For more information about `PVUIDXT`, see your iWay Server documentation.

Example Using the PVUIDXT Exit

The following example sets a group of user accounts to be trusted. If a user ID is part of the trusted group, a password is required regardless of whether the trusted option is set in the WebFOCUS Client.

Compilation and usage of this exit follows the standard methods for user subroutines, which is gencpgm on most platforms. Next, copy the exit to the user directory of EDACONF, or set the environment variable IBICPG to the name of the actual working directory. For more information, see the *iWay Stored Procedures Reference* manual.

```

/*
*****
*
* Program ID: pvuidxt.c
* Module ID : WinNT= pvuidxt.dll
*             UNIX = libpvuidxt.so or libpvuidxt.sl
*
*           This code is sample code, provided as-is.
*           No warranties are made as to its correctness or
*           suitability for any purpose
*
* Brief Desc: Sample X-PLATFORM WebFOCUS Reporting Server pvuidxt security
exit
*           This sample uses a simple input file, however the
*           sample could be easily modified to use LDAP or a DBMS.
*
* Input      : users.dat
*             (format of table)
*             {authuser,changeduser,password}
*
* Notes      : The password field is only needed for ophys that do
*             not support trusted nodes, i.e. NT
*             REVOKE may used in place of the changeduser
*             to not allow a user to connect to the WebFOCUS Reporting
Server.
*
* Author     : AJS (WebFOCUS Prodmgr)
* Date      : 00/11/29
* Maintenance Log:
*
*****
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include "pvuidxt.h"

void
#ifdef WIN32
    __declspec( dlllexport )

```

```

#endif
pvuidxt(
    char    *lpid    /* entity - reserved */
    ,char    *cmpid  /* IP Address */
    ,char    *uid    /* n byte user id */
    ,char    *pwd    /* n byte clear password */
    ,t_pvucb *ppvucb) /* Control Block */
{
    FILE    *fp;
    int     iLen,
           iUser;
    char    *pszUser,
           *pszFrom,
           *pszTo,
           *pszPswd,
           sep[]=" \n\r\0",
           szTable[120],
           szUserID[FULLENGTH_USERID_MAXLEN+1];

    /* ----- */
    /* Set default return code to OK, reset on failure */
    /* ----- */
    ppvucb->rc = PVUIDXT_SUCCESS;

    /* ----- */
    /* Copy userid and remove trailing spaces */
    /* needed for system function calls */
    /* ----- */
    memset(szUserID,'\0',FULLENGTH_USERID_MAXLEN+1); /* clear buffer */
    strncpy(szUserID,uid,FULLENGTH_USERID_MAXLEN);
    pszUser=strtok(szUserID,sep);

    /* ----- */
    /* Do not allow trusted connections */
    /* ----- */
    if(ppvucb->trusted_flag==PVUIDXT_TRUSTED)
    {
        ppvucb->rc=PVUIDXT_FAILURE;
        return;
    }

    /* ----- */
    /* user id was not supplied */
    /* ----- */
    if(pszUser==NULL)
    {
        ppvucb->rc=PVUIDXT_FAILURE;
        return;
    }

    /* ----- */
    /* Open security table */
    /* ----- */

```

```

if ((fp = fopen("pvuidxt.dat","r")) == NULL)
{
    ppvucb->rc=PVUIDXT_FAILURE;
    return;
}

/* ----- */
/* validate incoming user */
/* format of table is */
/* {authuser,changeduser,password} */
/* ----- */
while (fgets(szTable,120,fp)
{
    iLen=strlen(szTable);
    if(iLen>1)
    {
        pszFrom=strtok(szTable,sep);
        if(pszFrom!=NULL)
        {
            pszTo=strtok(NULL,sep);
            if(pszTo!=NULL)
                pszPswd=strtok(NULL,sep);

            iUser=strcmp(pszFrom,pszUser);
            if(iUser==0) /* match */
            {
                if(!strcmp("REVOKE",pszTo)) /* Access Denied */
                    ppvucb->rc=PVUIDXT_FAILURE;
                else
                {
                    strcpy(uid,pszTo);
                    if(pszPswd)
                        strcpy(pwd,pszPswd);
                    ppvucb->trusted_flag=PVUIDXT_TRUSTED; /* set user
to trusted */
                    ppvucb->rc =PVUIDXT_UPDATED; /* replace
userid */
                }
                break;
            }
        }
    }
}
fclose(fp);
return;
}

```

CHAPTER 5

WebFOCUS Administration

Topics:

- WebFOCUS Client Configuration Files
- Encrypting WFS Files
- IBIWEB.CFG Settings
- IBIDIR.WFS Settings
- SITE.WFS Settings
- Web Server Settings
- WebFOCUS Script (WFS) Commands

The WebFOCUS Client installation creates configuration files, several of which contain settings that you can use to customize WebFOCUS for your organization's environment. You can customize the behavior of WebFOCUS using the `ibiweb.cfg` file, and several WebFOCUS Script (WFS) files (`ibidir.wfs`, `site.wfs`, and `mime.wfs`). Additionally, you can use WFS commands in `site.wfs` (recommended) or `ibidir.wfs` to further customize processing and control of the WebFOCUS Client.

Important: In WebFOCUS Version 5 Release 2, all default global settings are located in `cgivars.wfs` instead of `ibidir.wfs`. Substitute `cgivars.wfs` for any references to `ibidir.wfs` settings that can be configured using the WebFOCUS Client Console. For more information about the WebFOCUS Client Console and any new settings in the `cgivars.wfs` file, see the *WebFOCUS Client Console* chapter in the *WebFOCUS Summary Of New Features*.

WebFOCUS Client Configuration Files

All WebFOCUS Client configuration files are installed in the edaconf directory, whose path is defined in ibiweb.cfg. The following table describes these configuration files.

File Name	Location of Files (Windows NT and Windows 2000)	Description
ibiweb.cfg	conf\web\cgi	The initial configuration file loaded by the CGI, ISAPI, or servlet. It is used to locate WFS files. For more information, see <i>IBIWEB.CFG Settings</i> on page 5-7. Note: When using WebFOCUS 4.3.5 or higher on Windows NT or Windows 2000, ibiweb.cfg replaces the Windows NT and Windows 2000 registry entries. On UNIX, ibiweb.cfg replaces the ibiweb.exe shell script.
edaserve.cfg	conf\bin	Used to locate binary dynamic link libraries (dlls). It is not used by WebFOCUS servlets.
ibstd.wfs	conf\etc	Loaded after ibiweb.cfg and is responsible for loading ibidir.wfs, cgivars.wfs, and mime.wfs. Do not edit this file. It is for internal use only.
ibidir.wfs	conf\etc	Contains installation default settings. This file can be edited, but it is preferable for customization to be performed in site.wfs. For more information, see <i>IBIDIR.WFS Settings</i> on page 5-9. When using WebFOCUS 4.3.5 or higher on Windows NT or Windows 2000, ibidir.wfs replaces the Windows NT and Windows 2000 registry entries.
cgivars.wfs	conf\etc	Do not edit this file. It is for internal use only.
mime.wfs	conf\etc	Contains information about available MIME types. For more information, see <i>Using the Redirect Setting in the MIME.WFS File</i> on page 5-5.
site.wfs	conf\etc	Contains site-specific settings and is used for customization. It can be used to override current settings and to create new settings. For more information, see <i>SITE.WFS Settings</i> on page 5-11.

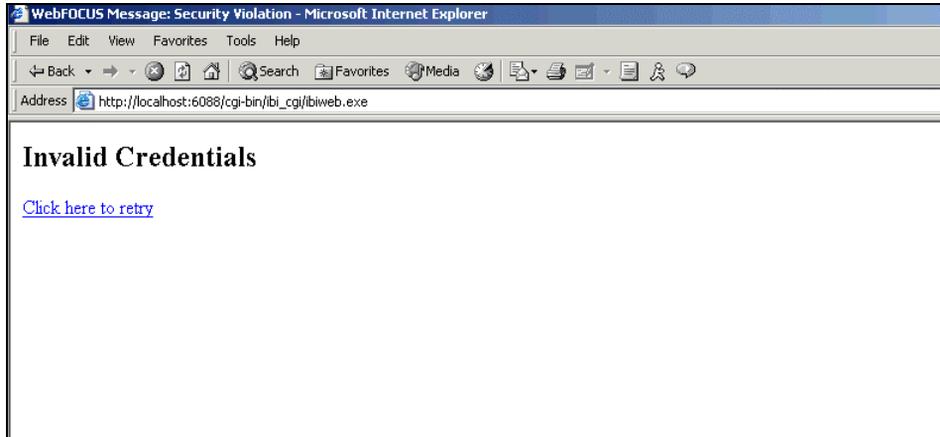
File Name	Location of Files (Windows NT and Windows 2000)	Description
mrchft.def	conf\etc	MRE only, defined in ibidir.wfs. Do not edit this file. It is for internal use only.
odin.cfg	conf\etc	This is the WebFOCUS communications file. It contains WebFOCUS Reporting Server nodes and is defined in ibidir.wfs.
default.wfs	conf\etc	Invokes a WebFOCUS Reporting Server request. Do not edit this file. It is for internal use only.
ibicommd.wfs	conf\etc	Contains WebFOCUS Reporting Server connection information. It is invoked by a WebFOCUS Reporting Server Signon, or from default.wfs.
vcp.html vcplnch.html vcplow.html vcpmain.html	conf\etc	Used for On-demand paging. Do not edit these files. They are for internal use only.
ib00000e.html	conf\etc	Contains WebFOCUS Client messages. Administrators can customize these messages. For more information, see <i>Using the ib00000e.html File to Customize WebFOCUS Messages</i> on page 5-4.

Example**Using the ib0000e.html File to Customize WebFOCUS Messages**

The following code displays an “Invalid Credentials” message and provides a link back to the WebFOCUS signon page (wfsignon.html):

```
<ERROR 00000>
:
</ERROR>
:
<ERROR 32033>
<ERROR 32034>
<HTML>
<HEAD><Title>WebFOCUS Message: Security Violation
</Title></Head>
<Body>
<H2>
Invalid Credentials
</H2>
<A HREF="wfsignon.html"
onClick="window.location='/ibi_html/wfsignon.html';return false">
Click here to retry</A>
</Body>
</HTML>
</ERROR>
```

Note: You can only modify the code between the HTML tags.



Example**Using the Redirect Setting in the MIME.WFS File**

The mime.wfs file contains information about format types available with WebFOCUS. You may edit the mime.wfs file (located in conf/etc) with extreme care. The rdr setting in this file can be used to control WebFOCUS behavior when a particular format type is used. For example, in the following sample mime.wfs file, if you are using a Windows browser with a MIME extension of .pdf, WebFOCUS will redirect report output to an Adobe Acrobat application.

```

<!          ext  mimeheader          format  rdr  hfs  trm  img  enc
<!
<ADDDTYPE> .html text/html          ascii  no  no  no  no  no
<ADDDTYPE> .htm  text/html          ascii  no  no  no  no  no
<ADDDTYPE> .htx  application/vnd.ms-excel  ascii  no  no  no  no  no
<ADDDTYPE> .xht  application/vnd.ms-excel  ascii  no  no  &astran  no  no
<ADDDTYPE> .wk1  application/vnd.ms-excel  binary  no  yes  no  yes  no
<ADDDTYPE> .txt  text/plain          ascii  no  no  no  no  no
<ADDDTYPE> .fex  text/plain          ascii  no  no  no  no  no
<ADDDTYPE> .sty  text/plain          ascii  no  no  no  no  no
<ADDDTYPE> .css  text/css          ascii  no  no  no  no  no
<ADDDTYPE> .orw  application/x-odin-int  binary  no  no  no  yes  no
<ADDDTYPE> .wp   text/plain          ascii  no  no  no  no  no
<ADDDTYPE> .hts  text/plain          ascii  no  no  no  no  no
<ADDDTYPE> .ftm  application/x-ftm        ascii  no  no  no  no  no
<ADDDTYPE> .gif  image/gif          binary  no  no  no  yes  no
<ADDDTYPE> .jpeg image/jpeg          binary  no  no  no  yes  no
<ADDDTYPE> .jpg  image/jpeg          binary  no  no  no  yes  no
<ADDDTYPE> .bmp  image/x-MS-bmp        binary  no  no  no  yes  no
<ADDDTYPE> .text text/plain          ascii  no  no  no  no  no
<ADDDTYPE> .tmp  text/plain          binary  no  yes  no  yes  no
<ADDDTYPE> .mas  text/plain          ascii  no  no  no  no  no
<ADDDTYPE> .xml  text/xml          ascii  no  no  no  no  no
<ADDDTYPE> .xls  application/vnd.ms-excel  binary  no  yes  no  yes  no
<ADDDTYPE> .ps   application/postscript  ascii  no  no  &astran  no  no
<ADDDTYPE> .prn  application/x-prn        ascii  no  no  &astran  no  no
<if> IBIMR_returntype contains "sync"
<ADDDTYPE> .doc  text/plain          ascii  no  no  no  no  no
<ELSE>
<ADDDTYPE> .doc  application/msword        ascii  no  yes  &astran  no  no
<ENDIF>
<IF> BROWSER EQ "MICROSOFT"
<ADDDTYPE> .pdf  application/pdf          ascii  yes  yes  &astran  no  no
<ADDDTYPE> .dif  application/x-dif        ascii  no  yes  &astran  no  no
<ENDIF>
<IF> BROWSER EQ "OTHER"
<ADDDTYPE> .pdf  application/pdf          ascii  no  yes  &astran  no  no
<ADDDTYPE> .dif  application/x-dif        ascii  no  yes  &astran  no  no
<ENDIF>

```

Encrypting WFS Files

Among the WebFOCUS configuration files are several WebFOCUS Script (WFS) files, which may be protected using the encrypt command. When files are encrypted, they are secure from unauthorized examination. You can decrypt an encrypted file to view it.

Syntax

How to Encrypt or Decrypt WFS Files

To encrypt or decrypt .wfs files, issue the following command from the WebFOCUS Client directory:

```
ibiweb.exe {-encrypt|-decrypt} password filename
```

where:

password

Is the password used to encrypt or decrypt the file.

filename

Is the name of the file you want to encrypt or decrypt.

Example

Encrypting and Decrypting IBIDIR.WFS

To encrypt ibidir.wfs, issue the following command from the WebFOCUS Client directory:

```
ibiweb.exe -encrypt mypassword d:\ibi\client51\conf\etc\ibidir.wfs
```

To decrypt ibidir.wfs, issue the following command from the WebFOCUS Client directory:

```
ibiweb.exe -decrypt mypassword d:\ibi\client51\conf\etc\ibidir.wfs
```

IBIWEB.CFG Settings

ibiweb.cfg is the initial configuration file loaded by the WebFOCUS CGI, ISAPI, or servlet. It is used to locate WFS files. It also, optionally, can be used to set tracing options, encrypt the WebFOCUS user cookie, and set the appropriate alias to load either the WebFOCUS Reporting Server or WebFOCUS Maintain CGI if the alias is different from the default.

The installation program only inserts edaconf into ibiweb.cfg. You must manually add all other settings.

Setting	Usage	Notes
<code>edaconf=directory</code>	<i>directory</i> Is the WebFOCUS Client configuration directory.	Used to locate WFS files. It internally sets CGI_BASE_DIR equal to edaconf/etc.
<code>trace={on off}</code>	<i>on</i> Turns tracing on. <i>off</i> Turns tracing off.	Used to enable and disable tracing. Tracing can be turned on or off at any time (the Web server does not need to be restarted). Every request generates a new trace file.
<code>trace_location=directory</code>	<i>directory</i> Specifies a directory where trace files are written.	Defaults to the current directory.
<code>ENCRYPT_WF_USER={YES NO}</code>	Setting this variable to YES encrypts the WebFOCUS user cookie (WF_USER).	The WF_USER cookie is not encrypted by default.
<code>WFS_alias=URI;URI,...</code>	<i>URI</i> Is the Web server alias for the WebFOCUS CGI. If WebFOCUS Maintain is not being used, a single forward slash (/) may be used instead of a specific URI to represent all requests.	Must be set if the CGI or ISAPI alias defined in the Web server for WebFOCUS is different from the following default: <code>/cgi-bin/ibi_cgi</code>

Setting	Usage	Notes
<code>WFM_alias=URI;URI,...</code>	<i>URI</i> Is the Web server alias for the WebFOCUS Maintain CGI.	Must be set if the CGI or ISAPI alias defined in the Web server for WebFOCUS Maintain is different from the following default: <code>/ibi_cgi</code>

Example

Configuring IBIWEB.CFG

The following example depicts configuration settings that might appear in `ibiweb.cfg`:

```
edaconf = c:\ibi\client51\conf
trace = on
trace_location = c:\temp
MR_PGM_FLAGS = -----Y
```

In this example, the configuration directory for the iWay Connector is `c:\ibi\client51\conf`. Tracing is enabled, and the resulting trace files are written to a specified temporary directory (`c:\temp`). A positioning flag is enabled, allowing a WebFOCUS callable exit to be enabled. For more information about the positioning flag, see *MR_PGM_FLAGS* on page 5-8.

MR_PGM_FLAGS

MR_PGM_FLAGS are positional flags that are set in `ibiweb.cfg`. These flags are used to enable WebFOCUS custom security exits (for details, see Chapter 4, *WebFOCUS Custom Security Exits*) and to turn off the prompting of variables in MRE.

Each position in the flag is either a `y` or a `-` (for default behavior). Filling in a position with a `y` activates the following behaviors:

Flag Setting	Usage
Positions 1-4, 6	Reserved for internal use.
Position 5	Enables the MRCEXT3 exit, which allows for an external encryption algorithm to be used for WebFOCUS encryption.
Position 7	Enables the MRCEXT4 exit, which allows for the external authentication of a user's credentials during an MRE signon.
Position 8	Enables the MRCEXT5 exit, which allows for the external management of users. MRCEXT5 is used in conjunction with the MRCEXT1 exit.
Position 9	Turns off the prompting of variables in MRE.
Position 10	Enables WebFOCUS CGI callable exits to be invoked.

Example**Setting MR_PGM_FLAGS**

To use the MRCEXT4 exit (position 7) and turn off variable prompting (position 9), code the following in ibiweb.cfg:

```
MR_PGM_FLAGS=-----y-y
```

To use the MRCEXT3 exit (position 5) along with the MRCEXT4 exit (position 7), code the following in ibiweb.cfg:

```
MR_PGM_FLAGS=----y-y
```

IBIDIR.WFS Settings

All default settings generated by installing WebFOCUS can be found in ibidir.wfs. The following settings in this file may be changed:

Settings	Description
IBI_DOCUMENT_ROOT	Location of the WebFOCUS installation directory.
IBI_HTML_DIR	Location of the root directory for HTML and Java language files.
CGI_TEMP_DIR	Location of the temporary directory.
_TempFileTimeout	Deletes files from the temporary directory. Files are deleted if they are older than the number of seconds specified.
ERROR_BASE_FILE	Location of the file that contains error messages.
IBIJAVADIR	Location of the JRE/JDK installation directory. It is not used by WebFOCUS servlets.
IBIJAVAPATH	Location of Java language execution statements. It is not used by WebFOCUS servlets.
ENABLE_SSG	Setting this variable to Y (the default) enables server-side graphics.
IBI_REPORT_USER	Default WebFOCUS Reporting Server user ID. Only used when a user ID (IBIC_user) is not supplied.
IBI_REPORT_PASS	Default WebFOCUS Reporting Server password. Only used when a user password (IBIC_pass) is not supplied.
IBI_REPORT_SERVER	Default WebFOCUS Reporting Server. Only used when a server (IBIC_server) is not supplied.
IBI_DEFER_USER	Default alternate deferred receipt server user ID. Only used when a deferred user ID (IBID_user) is not supplied.
IBI_DEFER_PASS	Default alternate deferred receipt server password. Only used when a deferred user password (IBID_pass) is not supplied.

Settings	Description
IBI_DEFER_SERVER	Default alternate deferred receipt server. Only used when a deferred server (IBID_server) is not supplied.
IBID_validate	Validate alternate deferred receipt server user ID.
_eda_config	Location of the iWay Connector communications configuration file.
EXPIRE_REPORTS	<p>The time when reports generated by the WebFOCUS Reporting Server are to be re-executed. The expiration time is set in seconds, and the default setting is 300 seconds.</p> <p>You can view a previously executed report on the browser without reexecuting its request. If the request for a report you are running is the same as the one executed previously (within 300 seconds), then the browser displays the previously executed report stored in the cache when you use the browser's Back, Reload, or Refresh button.</p> <p>You can set EXPIRE_REPORTS in the following ways:</p> <ul style="list-style-type: none"> To ensure that the reports are re-executed, set EXPIRE_REPORTS to 1 second. To view the browser's cached output without using the browser's Back, Reload, or Refresh button, set EXPIRE_REPORTS to a large number, such as 4,000,000,000 seconds.
WF_COOKIE_EXPIRATION	Number of minutes the WebFOCUS cookie remains in effect. A value of 0 means that there will be no expiration time applied to the cookie.
MR_BASE_DIR	Location of the MRE Repository.
MR_BASE_HTML_DIR	Location of the root directory of the MRE Repository.
MR_TEMPLATE_DIR	Location of HTML and GIF files for the MRE publish feature.
BKR_REPORT_DIR	FTP alias that is used to store ReportCaster files.
AGENT_NODE	ReportCaster's Agent Listener node.
AGENT_PORT	ReportCaster's Agent Listener port.
APPROOT	Location of the Application Namespace root directory.
APPADMIN	Location of the Administrator's user ID for the Application Namespace root directory.
IBIOPENURL	Used for PDA Sync, an optional mobile feature that is available with WebFOCUS.

SITE.WFS Settings

You can customize WebFOCUS for various behaviors, including enabling security exits (for more information, see Chapter 4, *WebFOCUS Custom Security Exits*), setting tracing and debugging options, executing code, and controlling cookie behavior. The following settings can be placed in `site.wfs` (recommended) or `ibidir.wfs` to customize WebFOCUS for your organization's environment.

Settings	Description
<code>MR_USER_ADMIN</code>	Enables the MRCEXT1 exit. This is not set by default.
<code>SAVE_FEX=filename</code>	Trace option that writes the output procedure being sent to the WebFOCUS Reporting Server. It contains only the packaged request and is used for debugging purposes only.
<code>_site_profile</code>	The following syntax enables you to include code to be executed on the WebFOCUS Reporting Server. <code>_site_profile=focuscommand</code> where: <code>focuscommand</code> Is any valid WebFOCUS Reporting Server syntax.
<code>_HTML_COMMENT_SITE</code>	Displays as an HTML comment, and is generally used for debugging purposes.
<code>URL_PROTOCOL</code>	Sets the default URL protocol. Set to <code>http</code> or <code>https</code> (if SSL is in use).
<code>REDIRECT_AUTODELETE</code>	Valid values are <code>ON</code> or <code>OFF</code> (the default). If <code>ON</code> , the redirected request is automatically deleted after being returned to the browser.
<code>IBI_COOKIE_DOMAIN</code> <code>IBI_COOKIE_PATH</code> <code>IBI_COOKIE_SECURE</code>	Control the behavior of the WebFOCUS cookie. For more information about these variables, see <i>WebFOCUS Cookie Variables</i> in Chapter 3, <i>WebFOCUS Security</i> .

Tip: You can use WFS commands with WFS variables in `site.wfs` (recommended) or `ibidir.wfs` to further customize processing and control of the WebFOCUS Client. For more information, see *WebFOCUS Script (WFS) Commands* on page 5-13.

Web Server Settings

You can use the following standard Web server settings in addition to your WFS settings (see *IBIDIR.WFS Settings* on page 5-9 and *SITE.WFS Settings* on page 5-11) for processing within your WFS files:

Web Server Setting	Description
AUTH_TYPE	Specifies the authentication scheme (basic or digest) if an authorization header is supplied.
CONTENT_LENGTH	Stores the number of bytes of data sent. For POST requests only.
CONTENT_TYPE	Designates the MIME type of attached data.
DOCUMENT_ROOT	Specifies the actual address of the host directory.
HTTP_ACCEPT	Specifies the media (MIME) type the WebFOCUS Client prefers to accept, separated by commas.
HTTP_USER_AGENT	Identifies the browser (or other client) making the request. It can be used to return content to different browsers.
HTTP_REFERER	Indicates the URL of the referring Web page.
PATH_INFO	Supplies any path information attached to the URL after the server address but before the query string.
PATH_TRANSLATED	The value of PATH_INFO with any virtual path name expanded into a directory specification.
QUERY_STRING	The information that follows the question mark (?) in the URL.
REMOTE_ADDR	The IP address of the client that made the request.
REMOTE_HOST	The fully-qualified Domain name of the client that made the request.
REMOTE_USER	If an authorization header was supplied and decoded by the server, the REMOTE_USER variable supplies the user ID. Note: In a Windows NT environment, &REMOTE_USER may be populated with the NT domain name in addition to the NT user ID, depending on the domain configuration. &WF_REMOTE_USER strips the domain name away for you.
REQUEST_METHOD	The HTTP request method.
SCRIPT_NAME	The name of the script program being run.
SERVER_NAME	The server's host name or IP address.

Web Server Setting	Description
<code>SERVER_PORT</code>	The TCP/IP port on which the request was received.
<code>SERVER_PROTOCOL</code>	The name and version of the information retrieval protocol that relates to a request.
<code>SERVER_SOFTWARE</code>	The name and version of the Web server.

Tip: You can use WFS commands with Web server variables to further customize processing and control of the WebFOCUS Client. For more information, see *WebFOCUS Script (WFS) Commands* on page 5-13.

WebFOCUS Script (WFS) Commands

You can use WebFOCUS Script (WFS) commands with WFS variables (see *IBIDIR.WFS Settings* on page 5-9 and *SITE.WFS Settings* on page 5-11) and Web server variables (see *Web Server Settings* on page 5-12) to further customize processing and control of the WebFOCUS Client.

You can use WFS commands to:

- Define, validate, and control WebFOCUS Client variables.
- Invoke WebFOCUS callable exits.
- Display HTML error or warning pages.

Tip: You can also use the following variables with WebFOCUS:

Form variables – set from an HTML page. These variables are automatically sent to the WebFOCUS Reporting Server. If a variable begins with *IBI* or *WF*, it is not automatically sent to the WebFOCUS Reporting Server.

Output variables – set by the WebFOCUS Reporting Server and passed back to the WebFOCUS Client. For more information about output variables, see *How to Use Variables to Affect WebFOCUS Reporting Server Output* on page 5-16.

Reference **WFS Command Syntax**

The following WFS commands can be placed in **site.wfs** (recommended) or **ibidir.wfs**.

Command Syntax	Description
<EXIT>	Causes the WebFOCUS Client to stop processing and exit immediately.
<HTMLFORM> <i>filename</i>	Displays an HTML page and allows for variable substitution in the HTML document. For more information, see <i>How to Display an HTML Page With Variables</i> on page 5-16.
<INCLUDE> <i>filename</i>	Separates WFS logic outside of the standard WFS files. The file must exist for processing to continue. For more information, see <i>How to Separate WFS Logic Outside of Standard WFS Files</i> on page 5-18.
<CONDITIONAL_INCLUDE> <i>filename</i>	Works in the same manner as <INCLUDE> <i>filename</i> except that the file does not need to exist. For more information, see <i>How to Separate WFS Logic Outside of Standard WFS Files</i> on page 5-18.
<SET> <i>variable_name(option)</i>	Allows for the validation and control of WebFOCUS Client variables passed from a cookie or form. For more information, see <i>How to Enable Validation and Control of Variables</i> on page 5-18.
<CALL> <i>function(param1,...)</i>	Invokes WebFOCUS callable exits. Each WebFOCUS callable exit can contain a maximum of 10 input parameters. For more information, see <i>How to Invoke a WebFOCUS Callable Exit</i> on page 5-19.
<IF> <i>variable operator value</i> <ELSE> <ENDIF>	Allows for the conditional checking of WebFOCUS Client variables. For more information, see <i>How to Allow for the Conditional Checking of Variables</i> on page 5-20.
<IFDEF> <i>variable</i> <ELSE> <ENDIF>	Checks for the existence of a WebFOCUS Client variable. For more information, see <i>How to Check for the Existence of a Variable</i> on page 5-21.
<IFNDEF> <i>variable</i> <ELSE> <ENDIF>	Checks to see if a WebFOCUS Client variable does not exist. For more information, see <i>How to Check to See If a Variable Does Not Exist</i> on page 5-22.

Command Syntax	Description
<pre><SENDVAR> name={constant &value} <ENSENDVAR></pre>	<p>Sends variables to the WebFOCUS Reporting Server for use with server procedures. It is used for variables that are not automatically sent to the WebFOCUS Reporting Server. For more information, see <i>How to Send Variables to the WebFOCUS Reporting Server</i> on page 5-22.</p> <p>Note: Information Builders recommends using the <code><SET> variable_name(option)</code> instead of the <code><SENDVAR></code> technique.</p>

Reference

WFS Language Syntax

When coding WFS commands, you can use the following syntax tools:

Syntax	Description
<pre><! > #</pre>	<p>Comment character that can be added at the beginning of any line. It informs the WebFOCUS Client that the current line is a comment.</p> <p>For example:</p> <pre><! This is a comment.> # This is also a comment.</pre>
<pre>\\=</pre>	<p>Continuation character that can be added at the end of any line. It informs the WebFOCUS Client that the next line is a continuation of the current line. For example:</p> <pre>long_string = this is a \\= very long string that \\= needs multiple lines</pre> <p>The following is an alternate concatenation technique:</p> <pre>long_string = this is a long_string =&long_string very long string that long_string =&long_string needs multiple lines</pre>
<pre>\n</pre>	<p>Carriage return line feed that enables you to put multiple commands on a single line.</p>

Syntax	Description
<code>value = {constant &variable}</code>	<p>Assigns a value to a variable in a WFS file as either a constant or a variable, where:</p> <p><i>constant</i> Is a literal value. If you include quotation marks, they are passed in as part of the variable.</p> <p><i>&variable</i> Is a placeholder for a value.</p>

Syntax**How to Use Variables to Affect WebFOCUS Reporting Server Output**

To affect the output of requests coming back from the WebFOCUS Reporting Server, run the following syntax on the WebFOCUS Reporting Server.

```
-TYPE WEBFOCUS CGIVAR varname=value
```

where:

varname

Is the name of the WebFOCUS Client variable.

value

Specifies the value of the WebFOCUS Client variable.

For more information about CGI variable names and possible values, see the *WebFOCUS Developing Reporting Applications* manual.

Syntax**How to Display an HTML Page With Variables**

The following command displays an HTML page

```
<HTMLFORM> filename
```

where:

filename

Is the name of the HTML file to be displayed.

The contents of the HTML document should use standard HTML syntax, and may include JavaScript. The following syntax may be added to the HTML document for variable substitution

```
<@-#echo var="variable"--> text
```

where:

variable

Is the WebFOCUS Client variable.

For a sample HTML document, see *Displaying an HTML Document* on page 5-17.

Example

Displaying an HTML Document

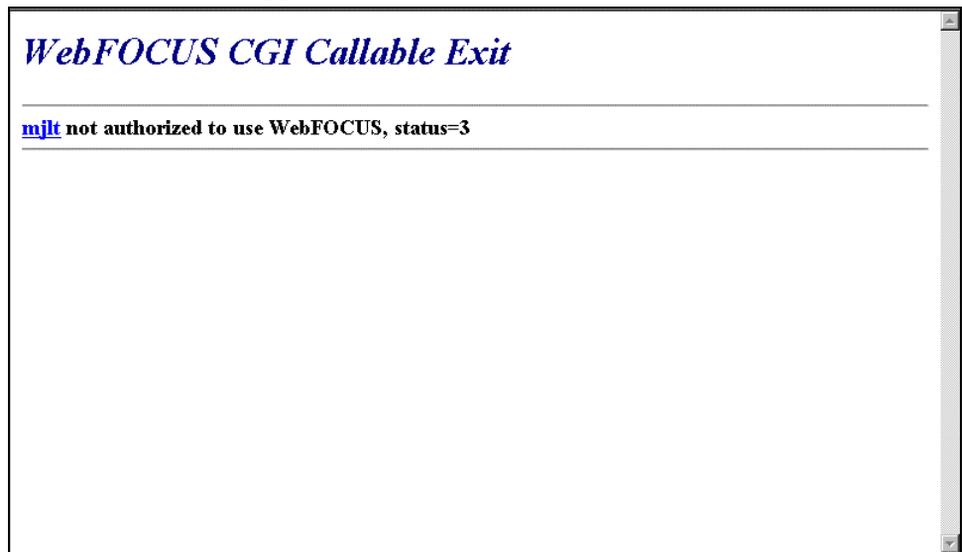
The following is the source code of HTML document wfexterr.htm, which displays an error page for a WebFOCUS callable exit:

```
<HTML>
<HEAD>
<TITLE>WebFOCUS External Security Exit</TITLE>
</HEAD>
<BODY>
<h1><i><font color="#000080">WebFOCUS CGI Callable Exit</font></i></h1>
<H3><HR>
<A HREF="/ibi_html/wfext.html"
onClick="window.location='/ibi_html/wfext.html';return false">
<@--#echo var="seccode"--></A> not authorized to use WebFOCUS, status=<@-
#echo var="RETCODE"-->
<HR></H3>
</BODY>
</HTML>
```

The following command displays wfexterr.htm:

```
<HTMLFORM> wfexterr.htm
```

This HTML document appears on the user's browser as follows:



Syntax

How to Separate WFS Logic Outside of Standard WFS Files

The following command separates WFS logic outside of standard WFS files. The file must exist for processing to continue.

```
<INCLUDE> filename
```

where:

```
filename
```

Is the name of the file to be included for WFS processing.

<CONDITIONAL_INCLUDE> filename works in the same manner as <INCLUDE> filename except that the file does not need to exist. For an example, see *Separating WFS Logic Outside of Standard WFS Files* on page 5-18.

Example

Separating WFS Logic Outside of Standard WFS Files

The following command checks for a file called wfext.wfs. If the file exists, it is included for WFS processing.

```
_exit=wfext.wfs
```

```
<CONDITIONAL_INCLUDE> &CGI_BASE_DIR&_DirSep&_exit
```

Syntax

How to Enable Validation and Control of Variables

The following <SET> command enables validation and control of WebFOCUS Client variables passed from a cookie or form

```
<SET> variable_name(option)
```

where:

```
variable_name
```

Is the name of the WebFOCUS Client variable.

```
option
```

Can be one of the following:

- pass - The variable becomes a Dialogue Manager amper variable on the WebFOCUS Reporting Server. This is the default.
- dontpass - The variable will not be passed to the WebFOCUS Reporting Server.
- protect - The variable cannot be set from the browser.
- number - The variable allowed can only be numeric.
- alpha - The variable allowed can only have alphanumeric characters [0-9, a-z, period (.), underscore (_), hyphen (-), exclamation point (!)].
- string - All characters are valid.
- length=n - Sets the maximum length of a variable in characters. The length is unlimited by default.

Note: The WebFOCUS Reporting Server has a variable limit of 12 characters.

For an example of the <SET> command, see *Using Alphanumeric Characters in IBIF_ex* on page 5-19.

Example**Using Alphanumeric Characters in IBIF_ex**

For non-MRE requests, only alphanumeric characters may be passed to IBIF_ex, which is the name of a WebFOCUS Reporting Server procedure. The following <SET> statement controls IBIF_ex so that it only accepts alphanumeric characters.

```
<if> IBIMR_drill EQ NULL AND IBIMR_action EQ NULL  
<SET> IBIF_ex(alpha)  
<endif>
```

Syntax**How to Invoke a WebFOCUS Callable Exit**

The following command invokes WebFOCUS callable exits

```
<CALL> function(parm1,parm2,...parm10)
```

where:

```
<CALL>
```

Is the command that invokes the WebFOCUS callable exit.

```
function
```

Defines the name of the actual function to be called (for example, *security*).

```
(parm1,parm2,...parm10)
```

Are the input parameters of the WebFOCUS callable exit. Each WebFOCUS callable exit can contain a maximum of 10 input parameters. The output buffer is not passed in as a parameter in this statement.

Note: The value of RETCODE can be checked to determine whether the exit function succeeded or failed.

For more detailed information about WebFOCUS callable exits, see Chapter 4, *WebFOCUS Custom Security Exits*. For an example, see *Using WFS Script Commands* on page 5-20.

Example Using WFS Script Commands

The following example makes use of a number of WFS scripting commands.

1. Check that variable *seccode* is passed in from the HTML form. If *seccode* is not passed in, an HTML error page is displayed and processing is stopped.

```
<ifndef> seccode
  <HTMLFORM> &IBI_HTML_DIR/wfext.html
  <EXIT>
<endif>
```

2. Invoke the WebFOCUS callable exit (called *security*).

```
<CALL> security(seccode)
```

3. Check for a valid return code. If the return code is valid, values are assigned to the WFS variable, and the <SENDVAR> command is issued to include variables in the WebFOCUS Reporting Server request. An invalid return code displays an HTML error page and exits.

```
<if> RETCODE EQ "0"
  IBIC_pass   = &wfepass
  IBIC_user   = &wfeuser
  IBIC_server = &wfenode
  <SENDVAR>
  WFUSER     = &wfeuser
  EMAIL      = &wfemail
  <ENDSENDVAR>
<else>
  <HTMLFORM> wfexterr.htm
  <EXIT>
<endif>
```

Syntax

How to Allow for the Conditional Checking of Variables

The following <IF> statement allows for the conditional checking of WebFOCUS Client variables

```
<IF> variable operator value
<ELSE>
<ENDIF>
```

where:

variable

Is any WebFOCUS Client variable.

operator

Is one of the following:

EQ, NE, CONTAINS, OR, AND.

value

Applies to any WebFOCUS Client variable or constant.

Note: By adding .upper to a WFS variable the value will be checked as if it has all upper case characters. This may be useful if you want to check the value that the user entered, without worrying about the case the user used.

For an example, see *Allowing for the Conditional Checking of Variables* on page 5-21.

Example**Allowing for the Conditional Checking of Variables**

The following <IF> statement can be used to ensure that the WebFOCUS logon page is invoked.

```
<IF> IBIC_user EQ NULL OR IBIC_user EQ ""
<HTMLFORM> &IBI_HTML_DIR/wfsignon.html
<EXIT>
```

The following <IF> statement ensures that any server value entered is not case sensitive.

```
<IF> IBIC_server.upper EQ "SERVER"
# INSERT YOUR CODE HERE...
<ENDIF>
```

The following <IF> statement checks whether or not constant .ibi.com is contained in the WebFOCUS Client variable HTTP_HOST.

```
<IF> HTTP_HOST contains ".ibi.com"
# INSERT YOUR CODE HERE...
<ENDIF>
```

Syntax**How to Check for the Existence of a Variable**

The following <IFDEF> statement checks for the existence of a WebFOCUS Client variable

```
<IFDEF> variable
<ELSE>
<ENDIF>
```

where:

variable

Is any WebFOCUS Client variable.

For an example, see *Checking for and Defining a Variable* on page 5-21.

Example**Checking for and Defining a Variable**

In the following example, if the variable *_ON_NT* exists, *PATH_SEP* is set to a semicolon (;). If the variable *ON_NT* does not exist, *PATH_SEP* is set to a colon (:).

```
<ifdef> _ON_NT
PATH_SEP=;
<else>
PATH_SEP=:
<endif>
```

Syntax

How to Check to See If a Variable Does Not Exist

The following <IFDEF> statement checks to see if a WebFOCUS Client variable does not exist

```
<IFDEF> variable  
<ELSE>  
<ENDIF>
```

where:

variable

Is any WebFOCUS Client variable.

For an example, see *Setting a Default Redirect Page* on page 5-22.

Example

Setting a Default Redirect Page

The following example sets a default redirect page when a valid WebFOCUS signon is achieved.

```
<ifndef> WF_SIGNON_MESSAGE  
WF_SIGNON_MESSAGE=/ibi_html/default.htm  
<endif>
```

Syntax

How to Send Variables to the WebFOCUS Reporting Server

The following command sends variables to the WebFOCUS Reporting Server for use with server procedures. It is used for variables that are not automatically sent to the WebFOCUS Reporting Server

```
<SENDVAR>  
name={constant/&value}  
<ENSENDVAR>
```

where:

name

Is a Dialogue Manager variable to be used by the WebFOCUS Reporting Server.

{*constant/&variable*}

Applies to any WebFOCUS Client variable.

where:

constant is a literal value. If you include quotation marks, they are passed in as part of the variable.

&variable is a placeholder for a value.

For an example, see *Sending User IDs to the WebFOCUS Reporting Server* on page 5-23.

Example

Sending User IDs to the WebFOCUS Reporting Server

The following example sends the WebFOCUS Reporting Server user ID to the WebFOCUS Reporting Server. It can be used from within any procedure.

```
<SENDVAR>  
CUSER=&IBIC_user  
<ENDSENDVAR>
```

Note: Information Builders recommends using the <SET> *variable_name(option)* instead of the <SENDVAR> technique.

CHAPTER 6

Tips and Techniques

Topics:

- Creating a WebFOCUS Site Profile
- Recommendations for Securing a WebFOCUS Web Site
- Minimum Recommended NTFS Permissions
- Application Security Integration
- Ensuring that the WebFOCUS Logon Is Invoked

The following topics provide WebFOCUS security tips and techniques, such as how to create WebFOCUS site profiles, and how to use WebFOCUS profile application security. It also provides recommendations for securing a WebFOCUS Web site, and the minimum NTFS permissions that should be set for the WebFOCUS Reporting Server and the WebFOCUS Client.

Creating a WebFOCUS Site Profile

Just as the WebFOCUS Reporting Server has a server profile (edasprof.prf) and users can have individual profiles (userid.prf), WebFOCUS itself can be configured with a profile.

A WebFOCUS site profile exists on the WebFOCUS Reporting Server and is executed immediately following edasprof.prf. While both profiles are FOCUS procedures, the WebFOCUS site profile has one significant advantage: FOCUS amper variables exported by the WebFOCUS Client with the sendvar command are available for use. This is in contrast with the edasprof.prf which cannot process amper variables.

Customers can use a WebFOCUS site profile to:

- Make a series of data source connections dependent on a variable sent from the WebFOCUS Client.
- Use a WebFOCUS Client variable (for example, &REMOTE_ADDR, &IBIMR_user) in a custom security procedure on the WebFOCUS Reporting Server, which can set other amper variables that affect subsequent report processing. This could be an example of application-based security.

Procedure

How to Enable a WebFOCUS Site Profile

1. In the ibdir.wfs file (or site.wfs), use `_site_profile` as follows:

```
_site_profile = -INCLUDE PROFILE
```

where:

```
PROFILE
```

Is the name of the site profile you create.

For more information about `_site_profile`, see Chapter 5, *WebFOCUS Administration*.

2. Create your profile procedure (or procedures) and place them in a directory where the WebFOCUS Reporting Server is configured to look for FOCUS procedures. You may put any FOCUS or Dialogue Manager commands in these profiles. Keep in mind that they execute after edasprof.prf and before the report procedure being sent by the WebFOCUS Client.

Note: IBIF_ex is generally used by self-service applications and IBIF_adhocfex is used by MRE, OLAP, and Developer Studio. Once you have completed these steps, the profile procedure (or procedures) will automatically execute; there is no need to restart or recycle anything.

Recommendations for Securing a WebFOCUS Web Site

The following section provides recommendations for securing a WebFOCUS Web site.

Disabling System Commands. You can disable system commands with the following setting in `edasprof.prf`:

```
SET OPSYSCMD=OFF
```

This does not affect filedefs, but some applications might need to use these system commands.

Note: You may also disable system commands by setting the `_site_profile` scripting variable. For example, the following command could be used in `ibidir.wfs` (or `site.wfs`):

```
_site_profile=SET OPSYSCMD=OFF
```

Disabling Ad Hoc Reports. The following command (in `ibidir.wfs` or `site.wfs`) disables ad hoc report capabilities:

```
<SET> IBIF_adhocfex(PROTECT)
```

The above command prevents users from sending ad hoc queries to the WebFOCUS Reporting Server.

Preventing Access to Stored Procedures. The following code can prevent a user from entering and executing a procedure with its fully qualified directory and file name:

```
<SET> IBIF_ex(ALPHA)
```

Directory Browsing. This setting should only be enabled for directories that require list functionality.

Guidelines for Setting File Permissions. General guidelines for setting file permissions on installed files are as follows:

- All files in the `ibi` directory structure should have a minimum of read access.
- Directories `ibi/srvrel/wfs`, `ibi/webfocusrel/basedir`, and `ibi/webfocusrel/temp` (where *rel* is the WebFOCUS release) should have the change permission (`rwed`).

These permissions apply to the effective user ID running WebFOCUS. Effective user IDs vary depending on the location and type of authentication being used. For example, if Web server authentication is not being utilized, the effective ID for file access on a typical Windows NT machine would be the anonymous or `IUSER_machinename`. This is the user ID that would need these `ibi` directory structure permissions.

Minimum Recommended NTFS Permissions

WebFOCUS Reporting Server. The minimum recommended NTFS permissions are:

Directory	Access of Effective User
\ibi\apps	Depends on the WebFOCUS components being used.
\ibi\srv436	Read, Execute
\ibi\srv436\home\	Read
\ibi\srv436\home\bin	Read, Execute
\ibi\srv436\home\catalog	Read
\ibi\srv436\home\copymgr	Read
\ibi\srv436\home\etc	Read
\ibi\srv436\home\nls	Read
\ibi\srv436\home\sample	Read
\ibi\srv436\home\system32	Read, Execute
\ibi\srv436\wfs	Read
\ibi\srv436\wfs\bin	Read
\ibi\srv436\wfs\catalog	Read
\ibi\srv436\wfs\dfm_dir	Depends on the WebFOCUS components being used.
\ibi\srv436\wfs\edatemp	Inherits from the parent directory.
\ibi\srv436\wfs\etc	Read
\ibi\srv436\wfs\fds	Depends on the WebFOCUS components being used.
\ibi\srv436\wfs\share	Inherits from the parent directory.
\ibi\srv436\wfs\tst	None
\ibi\srv436\wfs\user	Read, Execute
\ibi\srv436\wfs\web	None

Note: Only the WebFOCUS Reporting Server Administrator should have access to the inu.log file. Access should be denied to all other users.

Additional roles (such as a WebFOCUS Administrator) should be defined for WebFOCUS maintenance (adding or changing FOCUS procedures, Master Files), since all users do not need this type of unrestricted access.

To run an authenticating secure server, the built-in localsystem account has the following privileges by default:

- Act as part of the operating system.
- Replace a process level token.
- Increase quotas.

If you want to use an account other than localsystem, it must also have the above privileges. If these privileges are not set, then authentication will automatically be turned off.

WebFOCUS Client. User access should be set as follows:

Directory	Effective User	Access
ibi\WebFOCUS436\temp	Administrator	Full Control
ibi\WebFOCUS436\temp	WebFOCUS users	Read, Write
ibi\WebFOCUS436*	Everyone	Read
ibi\Client51*	Everyone	Read

Note:

- Depending on the type of access to the Web server you may choose not to use Everyone.
- Depending on the optional WebFOCUS products that are installed (MRE, ReportCaster), additional directories may require write privileges.

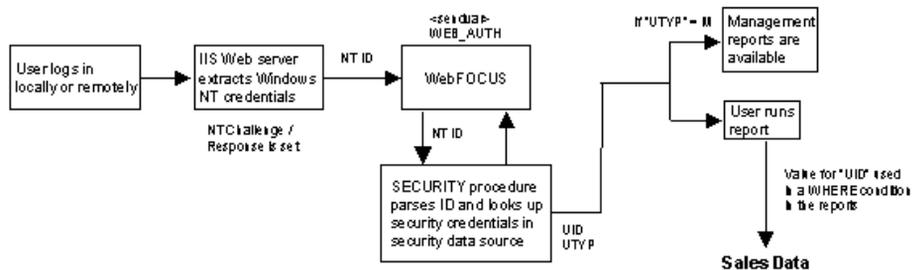
Application Security Integration

For this customer example, a secured WebFOCUS application enforces different levels of access to sales data depending on the user's position in a sales organization. Sales representatives can see their customer's data only, sales managers can see their territory and those of the representatives they manage, and regional sales managers and executive management can see everything.

This customer requested that their users should not have to authenticate themselves to WebFOCUS. Since WebFOCUS is on their intranet, users are already authenticated to the NT domain. Therefore, the customer chose the Microsoft® only option, from Web browser to network to Web server, and employed NT Challenge/Response on the Web server.

The customer created a WebFOCUS site profile called *security*. This customer's implementation is not compatible with ReportCaster since they are using NT Challenge/Response on the Web server.

Once each user is identified by this option, their application credentials are verified in a flat file security data source using a WebFOCUS procedure. Access to the flat file is controlled through operating system security. This flat file contains one record for each user: his or her last name (which is used in a WHERE condition for each report), and code that indicates whether or not the user is allowed to view management reports. The following diagram demonstrates this process:



Example Using WebFOCUS Profile Application Security

Add the following lines to site.wfs and enable basic authorization on your Web server:

```
<sendvar>
WEBID=&WF_REMOTE_USER
<endsendvar>
_site_profile=-INCLUDE SECURITY
```

When you run the request, the following code is run for every request issued to the WebFOCUS Reporting Server.

```
/*
*****
* SECURITY.FEX
*****
*/
FILEDEF SECDB DISK d:\ibi\srv436\wfs\catalog\secdb.txt
TABLE FILE SECDB
  PRINT REGION
  WHERE USERNAME EQ '&WEBID'
ON TABLE HOLD AS SECDATA
END
-RUN
-READ SECDATA &&AUTHREGION.A11.
```

The above code checks for &WEBID (the user ID passed from the <sendvar>) in the following table (called secdb.txt):

unxjlt	Midwest
wfuser	Northeast
bob	Southeast
mary	West
sally	Midwest

The following Master File describes secdb.txt:

```
FILE=SECDB, SUFFIX=FIX
SEGNAME=SECDB, SEGTYPE=S0
FIELD=USERNAME, ALIAS=E01, FORMAT=A20, $
FIELD=REGION, ALIAS=E02, FORMAT=A11, $
```

Ensuring that the WebFOCUS Logon Is Invoked

If the WF_COOKIE_EXPIRATION setting in ibidir.wfs is set to a time other than the default value, and the WebFOCUS cookie (WF_COOKIE) expires, then the following code can be used to ensure that the WebFOCUS logon window is invoked. The user is then reprompted for their security credentials. This technique can also be used to ensure that the WebFOCUS logon window is not bypassed.

Syntax

How to Ensure that the WebFOCUS Logon Is Invoked

```
<IF> IBIC_user EQ NULL OR IBIC_user EQ ""  
<HTMLFORM> c:\mydocs\wflagon.html  
<EXIT>
```

where:

IBIC_user

Is the WebFOCUS Reporting Server user ID.

c:\mydocs\wflagon.html

Is the logon window location.

For more information about the <HTMLFORM> command, see *How to Display an HTML Page With Variables* in Chapter 5, *WebFOCUS Administration*.

Index

Symbols

&IBIMR_user amper variable, 6-2
&MRUSER amper variable, 3-17 to 3-18
&REMOTE_ADDR amper variable, 6-2
&REMOTE_USER amper variable, 1-7, 3-12
&WF_REMOTE_USER amper variable, 3-12

A

ACCESS attribute, 2-3, 2-9, 2-15
access control, 1-3, 1-7, 2-2, 6-3 to 6-4
access to data, 1-1, 2-2
 commands, 2-10
 restricting, 1-3, 2-3, 2-12, 2-15
access to domains, 4-20
agent behavior, 3-2
AGENT_NODE setting, 5-9
AGENT_PORT setting, 5-9
aliases, 1-7
alphanumeric characters, 5-19
Already Verified Processing (AVP), 3-4
alternate cookie encryption exit (MRCEXT3), 4-29
amper variables, 1-7, 3-12, 3-17 to 3-18, 6-2
anonymous authentication, 1-6
APPADMIN setting, 5-9
application proxies, 1-5
application security integration, 6-6 to 6-7
APPROOT setting, 5-9
AUTH_TYPE setting, 5-12

authentication, 1-2
 logons, 3-8
 schemes, 1-6
AVP (Already Verified Processing), 3-4

B

basic authentication, 1-6
BKR_REPORT_DIR setting, 5-9

C

C header file for MRE callable exits, 4-17
CALL command, 5-13 to 5-14, 5-19 to 5-20
callable exits, 4-1, 5-13, 5-19
CGI (Communications Gateway Interface), 4-2
CGI_TEMP_DIR setting, 5-9
cgivars.wfs configuration file, 5-2
changing security credentials, 3-9
CHECK command, 2-10
COMBINE command and data security, 2-17 to 2-20
Communications Gateway Interface (CGI), 4-2
compiling a WebFOCUS CGI callable exit, 4-12
compiling MRE callable exits, 4-37
CONDITIONAL_INCLUDE command, 5-13 to 5-14, 5-18
confidentiality, 1-3, 1-8
configuration files, 5-1
configuring ibiweb.cfg, 5-8
CONTENT_LENGTH setting, 5-12
CONTENT_TYPE setting, 5-12

- cookies, 3-9
 - controlling expiration, 3-11
 - encrypting, 3-11
- CREATE command, 2-10
- creating a WebFOCUS CGI callable exit, 4-7
- creating a WebFOCUS servlet callable exit, 4-15
- creating the MRCEXT0 exit, 4-21
- creating the MRCEXT1 exit, 4-24 to 4-25
- creating the MRCEXT2 exit, 4-26 to 4-27
- creating the MRCEXT3 exit, 4-29 to 4-30
- creating the MRCEXT4 exit, 4-31
- creating the MRCEXT5 exit, 4-33 to 4-34
- custom signon forms, 3-16
- D**
- data access, 2-2, 2-12, 6-4
 - levels of, 2-9, 2-12
 - restricting, 1-3, 2-3, 2-15, 2-23
- data encryption, 2-22
 - performance considerations, 2-23
- data integrity, 1-3, 1-8
- data source security, 1-8, 2-1 to 2-4, 3-13
 - ACCESS attribute, 2-9
 - access control, 2-12, 2-23
 - access levels, 2-9, 2-12
 - central Master File, 2-17 to 2-18
 - CHECK FILE command and, 2-10
 - COMBINE command, 2-17 to 2-20
 - DBA attribute, 2-5
 - DBAFILE attribute, 2-17 to 2-18
 - encryption, 2-22, 2-24
 - filters, 2-20
 - identifying users, 2-6
 - integrating with WebFOCUS, 3-13
 - JOIN command, 2-17 to 2-20
 - data source security (*continued*)
 - passwords, 2-7 to 2-8
 - RESTRICT attribute, 2-12
 - restricting access, 2-13 to 2-16
 - special considerations, 2-4
 - storing DBA information centrally, 2-17 to 2-18
 - USER attribute, 2-6
- Database Administrator (DBA) security, 2-1
 - attributes, 2-2
 - passwords, 2-5, 2-15
- DBA (Database Administrator) security, 2-1
 - attributes, 2-2
 - displaying decision tables, 2-15
 - passwords, 2-5
- DBA exits, 4-1, 4-38, 4-40
 - calling parameters, 4-39
 - Master Files, 4-38
- DBAFILE attribute, 2-17 to 2-18
 - file naming requirements, 2-19
- DBATABLE procedure, 2-15
- decision tables, 2-15
- DECRYPT command, 2-10
- decrypting procedures, 2-24
- decrypting WFS files, 5-6
- default.wfs configuration file, 5-2
- deferred receipt security, 3-19
- DEFINE command, 2-10
- digest authentication, 1-6
- directory browsing, 1-8, 6-3
- disabling system commands, 6-3
- displaying HTML pages, 5-16
- DOCUMENT_ROOT setting, 5-12
- domain access, 4-20

E

eda_config setting, 5-9
edacnf setting, 5-7
edaserve.cfg configuration file, 5-2
edasprof.prf server profile, 6-2
enabling MRE callable exits, 5-8
 MRCEXT1 exit, 5-11
ENCRYPT command, 2-10, 2-22
ENCRYPT_WF_USER setting, 5-7
encryption, 1-3, 5-6
 cookies, 3-11
 data, 2-22
 Master Files, 2-22
 procedures, 2-24
 WFS files, 5-6
environment variables, 1-7, 5-7, 5-18, 5-22
ERROR_BASE_FILE setting, 5-9
EXIT command, 5-13 to 5-14
exits, 4-1
EXPIRE_REPORTS setting, 5-9
explicit authorization, 1-8
Explicit Verification Processing, 3-4

F

FIELD option to RESTRICT attribute, 2-11
field values, 2-15 to 2-16
 restricting access to, 2-15 to 2-16
fields, 2-13
 restricting access to, 2-13 to 2-14
filters, 1-5, 2-20
firewalls, 1-5
form variables, 5-13

G

generic proxies, 1-5
group permissions, 1-3

H

Header file for MRE callable exits, 4-17
HOLDSTAT parameter, 2-6
HTML form tags, 3-15
HTML pages, 5-16
 displaying, 5-17
HTML_COMMENT_SITE setting, 5-11
HTMLFORM command, 5-13 to 5-14, 5-16
HTTP variables, 3-9
HTTP_ACCEPT setting, 5-12
HTTP_COOKIE variable, 3-9
HTTP_REFERER setting, 5-12
HTTP_USER_AGENT setting, 3-9
HTTP_USER_AGENT variable, 5-12

I

ib00000e.html configuration file, 5-2, 5-4
IBI_DEFER_PASS setting, 5-9
IBI_DEFER_SERVER setting, 5-9
IBI_DEFER_USER setting, 5-9
IBI_DOCUMENT_ROOT setting, 4-23, 5-9
IBI_ENABLE_SSG setting, 5-9
IBI_HTML_DIR setting, 5-9
IBI_REPORT_PASS setting, 5-9
IBI_REPORT_SERVER setting, 5-9
IBI_REPORT_USER setting, 5-9
IBIC_pass parameter, 3-7, 3-9

IBIC_server parameter, 3-9
IBIC_user parameter, 3-7, 3-9
ibicommd.wfs configuration file, 3-4, 5-2
IBID_COOKIE_DOMAIN variable, 3-10
IBID_COOKIE_PATH variable, 3-10
IBID_COOKIE_SECURE variable, 3-10
IBID_pass parameter, 3-9
IBID_server parameter, 3-9
IBID_user parameter, 3-9
IBID_validate setting, 5-9
ibidir.wfs configuration settings, 5-2, 5-9
IBIJAVADIR setting, 5-9
IBIJAVAPATH setting, 5-9
IBIOPENURL setting, 5-9
ibiweb.cfg configuration settings, 5-2, 5-7 to 5-8
IBIWF_action variable, 3-8
IBMR_action command, 3-14
IBMR_random command, 3-14
ibstd.wfs configuration file, 5-2
identifying users, 1-2, 2-2
IF command, 5-20 to 5-21
IFDEF command, 5-13 to 5-14, 5-21
IFNDEF command, 5-13 to 5-14, 5-22
implicit authorization, 1-8
INCLUDE command, 5-13 to 5-14, 5-18
Integrated Windows Authentication (IWA), 1-6
invoking a WebFOCUS CGI callable exit, 4-11
invoking a WebFOCUS servlet callable exit, 4-16
invoking the WebFOCUS logon window, 6-8
IWA (Integrated Windows Authentication), 1-6

J

Java users, 4-20
JOIN command and data security, 2-17 to 2-20

L

LDAP (Lightweight Directory Access Protocol),
1-6, 3-6, 3-16, 4-1 to 4-2, 4-12, 4-22, 4-31, 4-34
Lightweight Directory Access Protocol (LDAP),
1-6, 3-6, 3-16, 4-1 to 4-2, 4-12, 4-22, 4-31, 4-34
logon authentication, 3-8
logon.htm signon form, 3-15
logond.htm signon form, 3-15
logons.htm signon form, 3-15

M

Managed Reporting Environment (MRE)
Administrator, 3-13
Managed Reporting Environment (MRE) Domain
Administrator, 3-13
Managed Reporting Environment (MRE) security,
3-1, 3-13
administrator account, 3-13
built-in features, 3-13
cookies, 3-14
logon forms, 3-14 to 3-15
reporting domains, 3-13
user capabilities, 3-13
user groups, 3-13
Master Files, 2-2, 4-38
encrypting, 2-22
MATCH command, 2-10
mime.wfs configuration settings, 5-2, 5-5
MR_BASE_DIR setting, 4-23, 5-9
MR_BASE_FILE setting, 4-23
MR_BASE_HTML setting, 4-23
MR_BASE_HTML_DIR setting, 5-9

- MR_COOKIE setting, 3-10, 3-14
 - MR_INFO setting, 3-10
 - MR_PGM_FLAGS setting, 5-7 to 5-9
 - MR_SIGNON action, 3-14
 - MR_TEMPLATE_DIR setting, 4-23, 5-9
 - MR_USER_ADMIN setting, 5-11, 5-13
 - MRCEXT module user-written exits, 4-1, 4-16
 - MRCEXT0 exit, 4-16, 4-20
 - compiling, 4-37
 - creating, 4-21 to 4-22
 - header file, 4-17
 - MRCEXT1 exit, 4-16, 4-23
 - compiling, 4-37
 - creating, 4-24 to 4-25
 - enabling, 5-11
 - header file, 4-17
 - setting variables and HTML form actions, 4-23
 - MRCEXT2 exit, 4-16, 4-25
 - compiling, 4-37
 - creating, 4-26 to 4-27
 - header file, 4-17
 - MRCEXT3 exit, 4-16, 4-29
 - compiling, 4-37
 - creating, 4-29 to 4-30
 - header file, 4-17
 - MRCEXT4 exit, 4-16, 4-30
 - compiling, 4-37
 - creating, 4-31
 - header file, 4-17
 - MRCEXT5 exit, 4-16, 4-20, 4-32
 - compiling, 4-37
 - creating, 4-33 to 4-34
 - header file, 4-17
 - mrchft.def configuration file, 5-2
 - MRE (Managed Reporting Environment)
 - Administrator, 3-13
 - MRE (Managed Reporting Environment) Domain Administrator, 3-13
 - MRE (Managed Reporting Environment) security, 3-1, 3-13
 - administrator account, 3-13
 - built-in features, 3-13
 - cookies, 3-14
 - logon forms, 3-14 to 3-15
 - reporting domains, 3-13
 - user capabilities, 3-13
 - user groups, 3-13
 - MRE callable exits, 4-1, 4-16, 4-23, 4-25, 4-30, 4-32
 - C header file, 4-17
 - compiling, 4-37
 - MRE internal variables, 3-17 to 3-18
 - MRE Repository security:, 3-19
 - MRE signon, 3-15
 - conditional, 3-17
 - customizing, 3-16
 - forms, 3-15
 - integrating with external security systems, 3-16
 - integrating with LDAP, 3-16
 - integrating with the WebFOCUS Reporting Server, 3-16
 - integrating with Web server, 3-16
 - MRE user IDs and passwords, 3-17 to 3-18
 - authenticating externally, 4-30
- ## N
- network layer security, 1-5, 3-12
 - NOPRINT option to RESTRICT attribute, 2-11
 - NTFS permissions, 6-4
- ## O
- odin.cfg configuration file, 5-2
 - operating system security, 1-4
 - output variables, 5-13, 5-16

P

- passwords, 2-2
 - authenticating, 3-4
 - changing, 2-5
 - setting externally, 2-23
- PATH_INFO setting, 5-12
- PATH_TRANSLATED setting, 5-12
- PDA Sync, 3-13
- PDC (Primary Domain Controller) server, 3-6
- physical security, 1-4
- pooled deployment, 3-4
- preference files, 4-20
- pre-started agents, 3-5
- Pre-Verify User ID Exit (PVUIDXT), 4-45 to 4-46
- Pre-Verify User ID Exit (PVUIDXT) function, 4-1, 4-43 to 4-44
- Primary Domain Controller (PDC) server, 3-6
- private deployment, 3-4
- procedure security, 2-24
- proxy gateways, 1-5
- PVUIDXT (Pre-Verify User ID Exit), 4-45 to 4-46
- PVUIDXT (Pre-Verify User ID Exit) function, 4-1, 4-43 to 4-44

Q

- QUERY_STRING setting, 5-12

R

- RDBMS security, 1-8
- read/write access, 2-9
- read-only access, 2-9, 2-15
- REBUILD command, 2-10
- redirect pages, 5-22
- REDIRECT_AUTODELETE setting, 5-11
- Relational Database Management Systems security, 1-8
- REMOTE_ADDR setting, 5-12
- REMOTE_HOST setting, 5-12
- REMOTE_METHOD setting, 5-12
- REMOTE_USER setting, 5-12
- ReportCaster security, 3-21
- RESTRICT attribute, 2-3, 2-10, 2-12
 - keywords, 2-10 to 2-11
- RESTRICT command, 2-10, 2-23
- restricting access, 2-13 to 2-14
 - to fields, 2-13 to 2-14
 - to segments, 2-13 to 2-14
- retfuncs exit function, 4-3 to 4-6
- runtime environment variables, 4-23
 - dynamically changing with MRCEXT1, 4-23

S

SAME option to RESTRICT attribute, 2-11

SAVE_FEX setting, 5-11, 5-13

SCRIPT_NAME setting, 5-12

Secure Sockets Layer (SSL), 1-8

securing a WebFOCUS Web site, 6-3

security, 6-1, 6-6

security attributes, 2-2, 2-21

security elements, 1-2

security exits, 4-1

security off setting, 3-2

security on setting, 3-2

security policy, 1-4

SEGMENT option to RESTRICT attribute, 2-11

segments, 2-13

- encrypting, 2-22
- restricting access to, 2-13 to 2-14

SENDVAR command, 5-13 to 5-14, 5-22 to 5-23

SERVER_NAME setting, 5-12

SERVER_PORT setting, 5-12

SERVER_PROTOCOL setting, 5-12

SERVER_SOFTWARE setting, 5-12

SET command, 5-13 to 5-14, 5-18 to 5-19

SET parameters, 2-6

- HOLDSTAT, 2-6
- PASS, 2-6 to 2-8
- USER, 2-6 to 2-8

setting file permissions, 6-3

setting passwords externally, 2-23

setting WebFOCUS communications defaults, 3-7

Shared Reports MRE exit (MRCEXT2), 4-25

signon forms, 3-15

signon processing, 3-8

site profiles, 6-2

site.wfs configuration settings, 5-2, 5-11

site_profile setting, 5-11, 5-13

SSL (Secure Sockets Layer), 1-8

T

TABLE command, 2-10

TCP/IP communications protocol, 4-43, 4-44 to 4-45

TempFileTimeout setting, 5-9

trace settings, 5-7

transin exit, 4-3 to 4-4

transout exit, 4-4 to 4-5

trusted security, 1-2, 1-8, 3-4

trusted security nodes, 4-43

- customizing, 4-43 to 4-45

U

update access, 2-9

URL_PROTOCOL setting, 5-9, 5-11

USER attribute, 2-6

user credentials, 3-9

- controlling expiration, 3-11

user IDs and passwords, 2-3, 3-7 to 3-8

- authenticating, 1-2, 3-4

user written security exits, 4-1

userid.prf profile, 6-2

V

VALUE option to RESTRICT attribute, 2-11, 2-15 to 2-16

values, 2-15

restricting access to, 2-15

variables, 5-13, 5-22

evaluating, 5-20 to 5-21

setting, 5-7

substituting, 5-16

vcp.html configuration file, 5-2

vcplnch.html configuration file, 5-2

vcplow.html configuration file, 5-2

vcpmain.html configuration file, 5-2

W

Web server security, 1-6, 6-6 to 6-7

access control, 1-7

authentication, 1-6

confidentiality, 1-8

data integrity, 1-8

environment variables, 1-7

integrating with WebFOCUS, 3-12

Web server settings, 5-12

WEBAUTH variable, 3-12

WebFOCUS administration, 5-1

WebFOCUS callable exits, 4-2, 4-12, 5-13, 5-19

WebFOCUS CGI callable exits, 4-1 to 4-2

compiling, 4-12

creating, 4-7

defining, 4-3, 4-5 to 4-6

implementing, 4-2

invoking, 4-11

populating WFEXT with retfuncs, 4-6

WebFOCUS Client configuration files, 5-2

WebFOCUS Client security, 3-1, 3-6

exits, 4-1

setting communications defaults, 3-7

WebFOCUS cookies, 3-9 to 3-10, 4-29
variables, 3-10

WebFOCUS custom security exits, 4-1
invoking, 5-13

WebFOCUS customization settings, 5-11

WebFOCUS DBA exits, 4-1, 4-38, 4-40
calling parameters, 4-39
Master Files, 4-38

WebFOCUS logon window, 6-8

WebFOCUS profile application security, 6-6 to 6-7

WebFOCUS Reporting Server input processing,
4-3 to 4-4

WebFOCUS Reporting Server output processing,
4-4 to 4-5

WebFOCUS Reporting Server security, 3-1 to 3-2

Administrator ID, 3-6

agent behavior, 3-2

connecting to, 3-7

deployment options, 3-4

edasprof.prf site profile, 6-2

exits, 4-1

processing modes, 3-4

setting communications defaults, 3-7

settings, 3-2

user IDs and, 5-23

variables and, 5-15, 5-22

WebFOCUS script (WFS) commands, 5-1, 5-13,
5-15, 5-18

WebFOCUS script (WFS) files, 5-1

encryption, 5-6

WebFOCUS security, 3-1, 6-1

WebFOCUS servlet callable exits, 4-1, 4-12

configuring, 4-13

creating, 4-15

defining, 4-14

invoking, 4-16

WebFOCUS signon processing, 3-8

WebFOCUS site profiles, 6-2

WebFOCUS Web sites, 6-3

- WEBID variable, 3-12
 - WF_alias setting, 5-7
 - WF_COOKIE setting, 3-9 to 3-10
 - encrypting, 3-11
 - WF_COOKIE_EXPIRATION setting, 3-11, 5-9
 - WF_SIGNON, 3-8, 3-14
 - WF_USER setting, 3-10
 - encrypting, 3-11
 - WF_VIEWER setting, 3-10
 - WFEXT module, 5-13, 5-19
 - WFEXT module user-written exits, 4-1 to 4-3, 4-5 to 4-6, 4-12 to 4-14
 - creating, 4-7, 4-15
 - implementing, 4-2
 - WFM_alias setting, 5-7
 - WFS (WebFOCUS script) commands, 5-13 to 5-14, 5-18
 - CALL, 5-14, 5-19 to 5-20
 - CONDITIONAL_INCLUDE, 5-14, 5-18
 - EXIT, 5-14
 - HTMLFORM, 5-14, 5-16
 - IF, 5-20 to 5-21
 - IFDEF, 5-14, 5-21
 - IFNDEF, 5-14, 5-22
 - INCLUDE, 5-14, 5-18
 - SENDVAR, 5-14, 5-22 to 5-23
 - SET, 5-14, 5-18 to 5-19
 - WFS (WebFOCUS script) files, 5-1
 - encryption, 5-6
 - WFS (WebFOCUS script) language syntax, 5-15
 - WFS (WebFOCUS script) logic, 5-18
 - WFS_alias setting, 5-7
 - Windows NT/2000 domain security, 3-6
 - write-only access, 2-9
- X**
- XDBSC module, 4-38

Reader Comments

In an ongoing effort to produce effective documentation, the Documentation Services staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please use this form to relay suggestions for improving this publication or to alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail: Documentation Services – Customer Support
Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

Fax: (212) 967-0460

E-mail: books_info@ibi.com

Web form: <http://www.informationbuilders.com/bookstore/derf.html>

Name: _____

Company: _____

Address: _____

Telephone: _____ Date: _____

E-mail: _____

Comments:

Reader Comments