z/OS

IBM

MVS Programming:
Product Registration

z/OS

MVS Programming:
Product Registration

> **Note**
>
> Before using this information and the product it supports, be sure to read the general information under "Notices" on page B-1.

Second Edition, September 2005

This is a major revision of SA22-7604-00.

This edition applies to Version 1 Release 7 of z/OS (5694-A01), Version 1 Release 7 of z/OS.e (5655-G52), and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this document, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States & Canada): 1+845+432-9405
FAX (Other Countries):
    Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: www.ibm.com/servers/eserver/zseries/zos/webqs.html

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:
- Title and order number of this document
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

About this document

Product registration services allow products to register with MVS when they are running on a particular system. Other products can then use registration services to determine what products are running on a particular system.

Product registration provides an additional function for optional products, or elements, of z/OS. These optional products, which can be either products, product features, or combinations of product and feature, can use registration services to determine, based on a policy the customer sets, whether they are enabled to run on a particular system.

This book describes how to use registration services.

Who should use this document

This book is for programmers who design and write, in assembler, C, or Java programs that use registration services. It requires an understanding of how to work with MVS system interfaces.

How to use this document

This book is one of the set of programming books for MVS. This set describes how to write programs in assembler language or high-level languages, such as C, FORTRAN, and COBOL. For more information about the content of this set of books, see *z/OS Information Roadmap*.

Note: If you call the services described in this book from assembler language programs, you must use a high-level assembler.

Where to find more information

Where necessary, this book references information in other books, using shortened versions of the book title. For complete titles, and order numbers of the books for all products that are part of z/OS, see *z/OS Information Roadmap*.

Information updates on the web

For the latest information updates that have been provided in PTF cover letters and Documentation APARs for z/OS™ and z/OS.e, see the online document at:
publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/ZIDOCMST/CCONTENTS.

This document is updated weekly and lists documentation changes before they are incorporated into z/OS publications.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM® messages you encounter, as well as for some system abends and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from these locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA™, and Clusters for AIX® and Linux™:

- The Internet. You can access IBM message explanations directly from the LookAt Web site at http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/.
- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations using LookAt from a TSO/E command line (for example: TSO/E prompt, ISPF, or z/OS UNIX® System Services).
- Your Microsoft® Windows® workstation. You can install LookAt directly from the *z/OS Collection* (SK3T-4269) or the *z/OS and Software Products DVD Collection* (SK3T4271) and use it from the resulting Windows graphical user interface (GUI). The command prompt (also known as the DOS > command line) version can still be used from the directory in which you install the Windows version of LookAt.
- Your wireless handheld device. You can use the LookAt Mobile Edition from http://www.ibm.com/servers/eserver/zseries/zos/bkserv/lookat/lookatm.html with a handheld device that has wireless access and an Internet browser (for example: Internet Explorer for Pocket PCs, Blazer or Eudora for Palm OS, or Opera for Linux handheld devices).

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from:

- A CD-ROM in the *z/OS Collection* (SK3T-4269).
- The *z/OS and Software Products DVD Collection* (SK3T4271).
- The LookAt Web site (click **Download** and then select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

Using IBM Health Checker for z/OS

IBM Health Checker for z/OS is a z/OS component that installations can use to gather information about their system environment and system parameters to help identify potential configuration problems before they impact availability or cause outages. Individual products, z/OS components, or ISV software can provide checks that take advantage of the IBM Health Checker for z/OS framework. This book refers to checks or messages associated with this component.

For additional information about checks and about IBM Health Checker for z/OS, see *IBM Health Checker for z/OS: User's Guide*. z/OS V1R4, V1R5, and V1R6 users can obtain the IBM Health Checker for z/OS from the z/OS Downloads page at http://www.ibm.com/servers/eserver/zseries/zos/downloads/.

SDSF also provides functions to simplify the management of checks. See *z/OS SDSF Operation and Customization* for additional information.

What Java level support is necessary for product registration

The product registration Java support requires that the following Java level or higher be installed:

- IBM SDK for z/OS Java 2 Technology Edition, Version 1.4 PTF UQ93743, product number 5655-I56.

Summary of changes

Summary of changes
for SA22-7604-01
z/OS Version 1 Release 7

This document contains information previously presented in *z/OS MVS JCL Reference*, SA22-7604-00, which supports z/OS Version 1 Release 1.

New information:
- "Registering and deregistering a product using Java" on page 3-5 describes the IFAEDJReg class which provides access to the z/OS product registration and deregistration services through Java™.

Changed information:
- References to OpenEdition have been replaced with z/OS UNIX System Services or z/OS UNIX.

Starting with z/OS V1R2, you may notice changes in the style and structure of some content in this document—for example, headings that use uppercase for the first letter of initial words only, and procedures that have a different look and format. The changes are ongoing improvements to the consistency and retrievability of information in our documents.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Summary of changes
for SA22-7604-00
z/OS Version 1 Release 1

This book contains information also presented in *OS/390 MVS Programming: Product Registration*.

This document contains terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

Chapter 1. Using Registration Services

Product registration provides a common mechanism for products to:
- Register (indicate that they are running) on a particular system
- Determine what products are registered (running) on a particular system

With z/OS, products, such as z/OS features, can use registration services to determine if they are enabled to run on a particular system. *z/OS MVS Product Management* describes product enablement, which requires that the product be defined appropriately in the enablement policy for the system.

The IFAPRDxx parmlib member contains the enablement policy, which the customer defines for a system.

Figure 1-1 shows how the product code, the enablement policy, and MVS registration services fit together to determine whether a product is enabled.

Product Code   MVS   Customer Policy

registers product name   product name   lists names of products

policy name

matches names
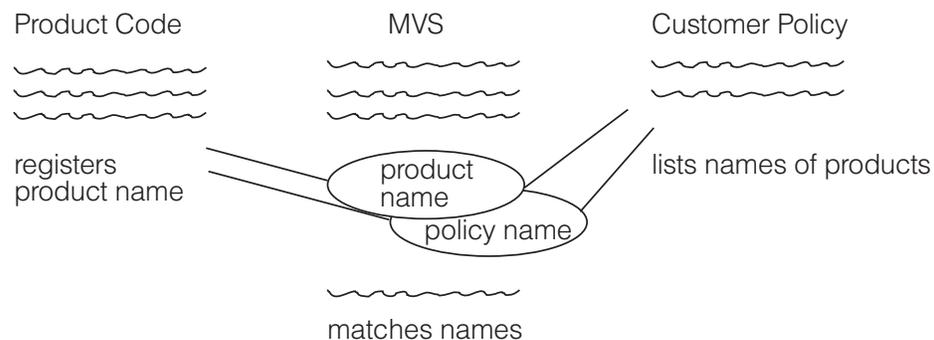
Figure 1-1. An Overview of Product Enablement

As Figure 1-1 shows, the product code issues the register request to indicate that it is running and check its enablement status. The customer policy in IFAPRDxx defines enablement status for products. When MVS processes the register request, it matches the product name definition in the request with the entries in the enablement policy to determine if the product is enabled on the system, then issues a return code to indicate enablement status. Based on the return code, the product continues to run or ends its processing.

If you are interested in how to enable a product, see:
- *z/OS MVS Product Management* for information about product enablement and reporting on registered products.
- *z/OS MVS Initialization and Tuning Reference* for an explanation of how to update IFAPRDxx.
- *z/OS and z/OS.e Planning for Installation* for a description of how to enable z/OS features.

This book, in contrast, describes how to use registration services. It is for the product programmer who needs to know:
- How to use registration services to register a product. See "Registering a Product" on page 1-2.
- How to use registration services to check product status — determine if a product is registered or enabled, or both, on a particular system. See "Checking Product Status" on page 1-3.

Registration services provide a standard mechanism for determining when a product is running or enabled on a system. Thus, the services are useful for all products with known dependencies on other IBM products or the products of independent software vendors or solution providers.

Examples of using the services appear in Chapter 3, "Examples," on page 3-1.

Registering a Product

To register a product, issue the Register service. See "Using the Register Service." When a product calls the Register service, MVS determines, based on the register request and the enablement policy defined in IFAPRDxx, whether or not the product is enabled to run on the system.

If the Register request type and the policy entry indicate that the product can run on the system, MVS registers the product as one that is running. Other products can then use the Query_Status and List_Status services to check whether your product is running. The system and other products assume that a product that is registered is a product that is running on a particular system.

Thus, it is important that, when your product finishes processing, it issue the Deregister service to tell MVS that it is no longer running. See "Using the Deregister Service" on page 1-3.

Using the Register Service

When a product issues the Register service, the system checks the enablement policy in IFAPRDxx. If the check is successful, MVS issues a return code of 0 and adds the product to its list of registered (running) products.

For the check to be successful, you need to select the type of Register request very carefully, depending on what you want to do:

- To register your product without regard to the enablement policy, select **Ifaedreg_Type_Required**. When it processes your request, the system does not check the enablement policy. Use this register request when you are registering only to allow other products:
  - To determine if your product is running.
  - To access information you provide through the **Features** parameter.
- To register your product and consider it enabled even if there is no entry in the policy, select **Ifaedreg_Type_Standard**. This type of request is useful when your product can be enabled without any user change to the policy in IFAPRDxx. With the standard register request, you get return code 4 (indicating that the product is disabled) only when there is a matching statement that explicitly disables the product.
- To register your product and consider it disabled when there is no entry in the policy, select **Ifaedreg_Type_NotFoundDisabled**. For the request to be successful, there must be a matching statement in the policy that explicitly enables the product. You get return code 4 when the product is explicitly disabled or when there is no matching statement.

The product definitions in the enablement policy can contain wildcard characters (? and *), and MVS allows wildcard matching so that a single policy statement can apply to multiple products.

Because of the interaction between the product definition in the register request, the type of register request, and the contents of IFAPRDxx, make sure that your

product documentation provides the information users need to update IFAPRDxx, as described in *z/OS MVS Initialization and Tuning Reference*.

The placement of the Register request in your product code is also important. Most products and separately orderable features would invoke the Register service during initialization. Products or features that have multiple entry points or that allow branch entry must consider registering at each possible point of invocation.

If other products need information about your product, you can use the **Features** parameter to pass the information. Callers of the Query_Status service can obtain this information, but you need to define its contents and format to enable the callers to interpret the information correctly.

See "Register Service (IFAEDREG)" on page 2-2 for a complete description of the service, including the various types of register request.

Using the Deregister Service

While the system can automatically deregister a product during task or address space termination, it is a good practice to issue the Deregister service when a registered product completes its processing.

Issuing the Deregister service ensures that any status queries that other products issue return correct results. The system considers a registered product to be a running product. If your product stops running but does not deregister, any query of its status will indicate that it is still running.

See "Deregister Service (IFAEDDRG)" on page 2-9 for a complete description of the service.

Checking Product Status

There are two services you can use to check product status:
- Query_Status, described in "Query_Status Service (IFAEDSTA)" on page 2-11
- List_Status, described in "List_Status Service (IFAEDLIS)" on page 2-15

Which service you need depends on the information your product requires:
- To determine if a specific product is registered and obtain its enablement status (enabled, disabled, or not known), issue the Query_Status service.
- To obtain information about the registration and enablement status of one or more products, issue the List_Status service.
- To determine what entry in the enablement policy the system would use to determine the enablement status of a particular product, issue the List_Status service.

Both services return information in data areas mapped for the assembler language programmer in mapping macro IFAEDIDF and for the C programmer in include file IFAEDC.

Before you issue either service, you need to know how any product you are interested in was defined when it was registered.

If you are using Query_Status to request the status of a specific product, you might need additional documentation from the product. When a product registers, it can provide information for the system to pass to the caller of Query_Status. If you are

interested in a product that provides this additional information, you need to understand the content and format of the information you will receive.

Chapter 2. Coding Registration Services

There are four registration services:

- Register service — registers a product or feature with MVS
- Deregister service — deregisters a product, usually done when an element completes processing.
- Query_Status service — checks the status of a specific product
- List_Status service — checks the status of one or more products

These callable services share common invocation characteristics and common processing considerations.

Invoking the Services

The following information describes the environment required, restrictions, register information, performance implications, and abend codes for the registration services.

Environment

The environment for the callers are:

Minimum authorization:	Problem state and any PSW key
Dispatchable unit mode:	Task
Cross memory mode:	PASN=HASN=SASN
AMODE:	31-bit
ASC mode:	Primary
Interrupt Status:	Enabled for I/O and external interrupts
Locks:	No locks may be held.
Control parameters:	Control parameters must be in the primary address space.

Programming Requirements

- If you are coding in assembler, include mapping macro IFAEDIDF. It provides return code equates for the various services and mappings for the output from the Query_Status service and the List_Status service. For a description of IFAEDIDF, see *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*.
- If you are coding in C, include file IFAEDC provides data definitions for the various services. For a description of IFAEDC, see "IFAEDC" on page 2-21.
- If you are coding in Java, use the methods in the IFAEDJReg class. See "Registering and deregistering a product using Java" on page 3-5 for more information.

Restrictions

- The caller cannot have an established FRR.
- An unauthorized caller of the Register service cannot register if there are already 10 successful registrations (counting all products) made by unauthorized callers from that address space.

Register Service (IFAEDREG)

- An unauthorized caller cannot deregister a product that was registered by an authorized caller.
- An unauthorized caller cannot deregister a product that was registered from another address space.

Input Register Information

Before issuing any registration service, the caller does not have to place any information into any register unless using it in register notation for a particular parameter, or using it as a base register.

Output Register Information

When control returns to the caller, the general purpose registers (GPRs) contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

When control returns to the caller, the ARs contain:

Register	Contents
0-1	Used as work registers by the system
2-13	Unchanged
14-15	Used as work registers by the system

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of register on which the caller depends, the caller must save them before issuing the service, and restore them after the system returns control.

Performance Implications

These services should not be used in a performance-sensitive environment.

ABEND Codes

Callers of the registration services might encounter the following ABEND codes:

0C4	**Meaning**: The system cannot properly access a user-provided parameter.
B78	**Meaning**: The caller was not enabled for I/O and external interrupts.

Register Service (IFAEDREG)

Use the Register service (IFAEDREG) to register a product with MVS. You can register a product or a unique product/feature combination. When you register a product with MVS, you indicate that the registered product is running on the system.

The Register service returns information to the caller and also maintains information that other callers can query to determine if products are registered (running) and enabled on the system.

If the product is an optional z/OS element, feature, or element/feature combination, MVS can also determine whether the element is enabled on this system.

To determine enablement, the system matches the product identified in the call against the policy statements in parmlib member IFAPRDxx.

It is possible, because of wildcard characters (? and *) in the policy statements, that multiple policy statements might match the given input product. In that case, MVS uses the ″best″ match to determine whether or not the product is enabled, using the following rules:

1. An exact match is better than a wildcard match. There is no differentiation between two wildcard matches.

2. The parameters are processed in the following order: Prodowner, ProdID, Prodname, Featurename, Prodvers, Prodrel, and Prodmod. An exact match on a parameter earlier in the list (such as Prodowner) is better than a match on a parameter later in the list (such as Prodname).

3. If, after applying the first two rules, more than one match remains, MVS uses the first match of those that remain.

If product code is neither in supervisor state nor running under a system key, it cannot issue more than 10 register requests.

Syntax

```
CALL  IFAEDREG,              (Type
                             ,Prodowner
                             ,Prodname
                             ,Featurename
                             ,Prodvers
                             ,Prodrel
                             ,Prodmod
                             ,ProdID
                             ,Featureslen
                             ,Features
                             ,Prodtoken
                             ,Returncode)
```

In C: the syntax is similar. You can use either of the following techniques to invoke the service:

1. `ifaedreg (Type,...Returncode);`
   When you use this technique, you must link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB.

2. `ifaedreg_byaddr (Type,...Returncode);`
   This second technique requires AMODE=31, and, before you issue the CALL, you must verify that the IFAEDREG service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

In Assembler: Link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB unless you use either of the following techniques as an alternative to CALL IFAEDREG:

1. `LOAD EP=IFAEDREG`
   `Save the entry point address`
   `...`

```
        Put the saved entry point address into R15
        Issue CALL (15),...

2.  L   15,X'10'                Get CVT
    L   15,X'8C'(,15)           Get ECVT
    L   15,X'1C0'(,15)
    L   15,4(,15)
    L   15,0(,15)               Get address of IFAEDREG
    CALL (15),(...)
```

Both of these techniques require AMODE=31. If you use the second technique, before you issue the CALL, you must verify that the IFAEDDRG service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

In Java: Use the methods in the IFAEDJReg class. Prior to invoking your application, the ifaedjreg.jar file must be available on the application's classpath and the registration native library must be available on the application's libpath. See "Registering and deregistering a product using Java" on page 3-5 for more information.

Parameters

Type

Supplied parameter:
*   Type: Integer
*   Length: Full word

Type identifies the type of register request. The field must contain a value that represents one or more of the possible types. You add the values to create the full word. Do not specify a type more than once. The possible types, and their meanings, are:

Ifaedreg_Type_Standard

The system is to register the product, check the enablement policy, and issue a successful return code unless the product is explicitly disabled in the policy. If the product is explicitly disabled, the system does not register the product and does issue return code 4. If you want the service to issue return code 4 (Ifaedreg_Disabled) when the product is not found in the policy, specify Ifaedreg_Type_NotFoundDisabled.

Ifaedreg_Type_Required

The system is to register the product but not check the enablement policy. Use this option when registering solely for status queries. Because the system does not check the enablement policy, you cannot get return code 4 (Ifaedreg_Disabled).

Ifaedreg_Type_NoReport

The system is to register the product but not report the product in the software registration report or the response to a DISPLAY command (unless the command specifies ALL). You might use this option when registering solely for status queries. Because the system does not check the enablement policy, you cannot get return code 4 (Ifaedreg_Disabled).

Ifaedreg_Type_LicensedUnderProd

The system is to register the product/feature combination, but the product/feature combination cannot be ordered separately. The software registration report will differentiate this type of registration from others; a person looking at the report can easily tell that there is no need to check the ordering information for this product/featurename combination.

Ifaedreg_Type_DisabledMessage

The system, if it finds the product to be disabled, is to issue message

IFA104I, described in *z/OS MVS System Messages, Vol 8 (IEF-IGD)*. Thus, the caller does not have to issue the message. The system issues message IFA104I with no console ID specified, and with routing codes 10 (System/Error Maintenance) and 11 (Programmer Information).

Ifaedreg_Type_NotFoundDisabled
The system, if it does not find the product in the enablement policy, is to treat the product as disabled rather than enabled. That is, if the product is not found, the system does not register the product and does issue return code 4 (Ifaedreg_Disabled). If you also specify Ifaedreg_Type_DisabledMessage, the system issues message IFA104I. For a description of this message, use LookAt or see *MVS System Messages*. For a description of LookAt, see "Using LookAt to look up message explanations" on page vii.

,Prodowner
Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Prodowner specifies the name of the product owner (vendor). IBM products, for example, always use IBM CORP or IBM_CORP.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Prodname
Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Prodname specifies the name of the product.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Featurename
Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Featurename specifies the name of the feature within the product or blanks if there is no feature name.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Prodvers

Supplied parameter:
- Type: EBCDIC
- Length: 2 bytes

Prodvers specifies the product version identification or blanks if there is no version identification.

The characters can be upper-case or lower-case alphabetics and numerics. You can use embedded blanks.

The system performs all comparisons in upper case.

If the version identification is less than 2 bytes, left-justify it in the field and pad it on the right with EBCDIC blanks.

,Prodrel

Supplied parameter:
- Type: EBCDIC
- Length: 2 bytes

Prodrel specifies the product release identification or blanks if there is no release identification.

The characters can be upper-case or lower-case alphabetics and numerics. You can use embedded blanks.

The system performs all comparisons in upper case.

If the release identification is less than 2 bytes, left-justify it in the field and pad it on the right with EBCDIC blanks.

,Prodmod

Supplied parameter:
- Type: EBCDIC
- Length: 2 bytes

Prodmod specifies the product modification level or blanks if there is no modification level.

The characters can be upper-case or lower-case alphabetics and numerics. You can use embedded blanks.

The system performs all comparisons in upper case.

If the modification level is less than 2 bytes, left-justify it in the field and pad it on the right with EBCDIC blanks.

,ProdID

Supplied parameter:
- Type: EBCDIC
- Length: 8 bytes

ProdID specifies the product identifier. IBM products, for example, use the product's program number.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 8 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Featureslen
Supplied parameter:
- Type: Integer
- Range: 0-1024
- Length: Full word

Featureslen specifies the length of the features parameter that follows.

,Features
Supplied parameter:
- Type: Character (EBCDIC recommended)
- Length: 1-1024 bytes

Features contains any information that you want the system to pass to the caller of the Query_Status service. (Featureslen specifies the length of the information.)

If you do not need to pass information to callers of the Query_Status service, code 0 in the Featureslen parameter. The system then ignores the contents of the Features parameter, but the service syntax requires that you supply a value.

If you do need to pass information to the callers of Query_Status, using EBCDIC can simplify the parsing requirements for the caller, but you do need to provide a mapping of the information for the caller to use. An alternate approach is to set up self-defining features information (such as `feature1=value1,feature2=value2,...`). This approach has the advantage of simplicity, but does use more system (common) storage.

If the product you are registering is already registered, the features information you specify here will replace the features information provided on any previous call, but only for the length provided on the previous call. For example, if the previous call specified a Featureslen of 16, and this call specifies 32, the system uses only the first 16 bytes of features information from this call.

,Prodtoken
Returned parameter:
- Type: Character
- Length: 8 bytes

Prodtoken contains the token the system returns to identify this particular registration. Save this token to supply as input to the Deregister service.

,Returncode
Returned parameter:
- Type: Integer
- Length: Full word

Returncode contains the return code from the Register service.

Return Codes

When the Register service returns control to the caller, Returncode contains the return code. To obtain the equates for the return codes:
- If you are coding in assembler, include mapping macro IFAEDIDF, described in *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*.
- If you are coding in C, use include file IFAEDC. See "IFAEDC" on page 2-21.

Register Service (IFAEDREG)

The following table describes the return codes, shown in decimal.

Return Code (decimal)	Equate Symbol Meaning and Action
00	**Equate Symbol**: IFAEDREG_SUCCESS **Meaning**: The product/feature combination is enabled and is permitted to execute. Note that, unless you request option Ifaedreg_Type_NotFoundDisabled, you will get this return code when the system does not find a policy statement that matches the product. **Action**: Proceed with normal execution.
04	**Equate Symbol**: IFAEDREG_DISABLED **Meaning**: The product/feature combination is **not** enabled; it is explicitly disabled and is not permitted to execute. To get this return code when the system does not find a policy statement that matches the product, you must also request option Ifaedreg_Type_NotFoundDisabled. **Action**: 1. Write the appropriate termination message to the terminal or log, unless the operator message issued because you requested Ifaedreg_Type_DisabledMessage provides enough information. 2. Set a return code to indicate termination for 'not ordered or not permitted to run' condition. 3. Terminate requestor's use of program.

08	**Equate Symbol**: IFAEDREG_NOTAVAILABLE
	
	Meaning: Environmental error: The Register service is not available on this system.
	
	Action:
	• If this version of the program must execute on a system that provides registration services:
	  1. Write the appropriate termination message to the terminal or log.
	  2. Set a return code to indicate termination because registration services are not available on this system.
	  3. Terminate requestor's use of program.
	• If this version of the program does not need to execute on a system that provides registration services, take the actions appropriate for the product/feature when you cannot determine if it is enabled.

12	**Equate Symbol**: IFAEDREG_LIMITEXCEEDED **Meaning**: Environmental error: This request exceeds the limit of 10 register requests by an unauthorized caller in this address space. **Action**: Use the Deregister service to remove unneeded registrations.
16	**Equate Symbol**: IFAEDREG_NOTTASKMODE **Meaning**: User error: The service was not called in task mode. **Action**: Avoid calling in this environment.
20	**Equate Symbol**: IFAEDREG_XM **Meaning**: User error: The service was called in cross-memory mode but requires PASN=HASN=SASN. **Action**: Avoid calling in this environment.
24	**Equate Symbol**: IFAEDREG_BADFEATURESLEN **Meaning**: User error: The Featureslen parameter was not in the range 0-1024. **Action**: Correct the parameter.

Return Code (decimal)	Equate Symbol Meaning and Action
28	**Equate Symbol**: IFAEDREG_NOSTORAGE **Meaning**: Environmental error: The system could not obtain the storage it needed to satisfy the request. **Action**: Contact the system programmer.
32	**Equate Symbol**: IFAEDREG_BADTYPE **Meaning**: User error: The type parameter did not specify a word formed from any combination of Ifaedreg_Type_Standard, Ifaedreg_Type_Required, Ifaedreg_Type_NoReport, Ifaedreg_Type_LicensedUnderProd, Ifaedreg_Type_DisabledMessage, and Ifaedreg_Type_NotFoundDisabled. **Action**: Correct the parameter.
36	**Equate Symbol**: IFAEDREG_LOCKED **Meaning**: User error: The service was called while holding a system lock. **Action**: Avoid calling in this environment.
40	**Equate Symbol**: IFAEDREG_FRR **Meaning**: User error: The service was called while having a functional recovery routine (FRR) established. **Action**: Avoid calling in this environment.

Deregister Service (IFAEDDRG)

Use the Deregister service (IFAEDDRG) to indicate that a registered product or product/feature combination is ending its processing. When a product registers with MVS, it indicates that it is running on the system. When it ends, the product issues the Deregister service to indicate that it has finished processing.

A product that issues the Register service receives a token that identifies the unique instance of the product. To deregister, the product calls the Deregister service and supplies the token. Note that the system automatically deregisters the product on termination of:

- The cross-memory resource owning task (TCB address in ASCBXTCB) that was active when the register request was done
- The address space that was the home address space when the register request was done.

If the product code is neither in supervisor state nor running under a system key, there are limitations on the use of Deregister:

1. You cannot deregister a product that was registered by a caller in supervisor state or running under a system key.
2. You can deregister only a product that was registered from your home address space.

Syntax

```
CALL  IFAEDDRG,              (Prodtoken
                             ,Returncode)
```

Deregister Service (IFAEDDRG)

In C: the syntax is similar. You can use either of the following techniques to invoke the service:

1. `ifaeddrg (Type,...Returncode);`
   When you use this technique, you must link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB.

2. `ifaeddrg_byaddr (Type,...Returncode);`
   This second technique requires AMODE=31, and, before you issue the CALL, you must verify that the IFAEDDRG service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

In Assembler: Link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB unless you use either of the following techniques as an alternative to CALL IFAEDDRG:

1. ```
   LOAD EP=IFAEDDRG
   Save the entry point address
   ...
   Put the saved entry point address into R15
   Issue CALL (15),...
   ```

2. ```
   L   15,X'10'                Get CVT
   L   15,X'8C'(,15)           Get ECVT
   L   15,X'1C0'(,15)
   L   15,4(,15)
   L   15,4(,15)               Get address of IFAEDDRG
   CALL (15),(...)
   ```
   Both of these techniques require AMODE=31. If you use the second technique, before you issue the CALL, you must verify that the IFAEDDRG service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

In Java: Use the methods in the IFAEDJReg class. Prior to invoking your application, the ifaedjreg.jar file must be available on the application's classpath and the registration native library must be available on the application's libpath. See "Registering and deregistering a product using Java" on page 3-5 for more information.

Parameters

Prodtoken
> Supplied parameter:
> - Type: Character
> - Length: 8 bytes
>
> Prodtoken contains the token the system returned when the product issued the Register service.

,Returncode
> Returned parameter:
> - Type: Integer
> - Length: Full word
>
> Returncode contains the return code from the Deregister service.

Return Codes

When the Deregister service returns control to the caller, Returncode contains the return code. To obtain the equates for the return codes:

- If you are coding in assembler, include mapping macro IFAEDIDF, described in *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*.
- If you are coding in C, use the include file IFAEDC. See "IFAEDC" on page 2-21.

The following table describes the return codes, shown in decimal.

Return Code (decimal)	Equate Symbol Meaning and Action
00	**Equate Symbol**: IFAEDDRG_SUCCESS **Meaning**: The product/feature combination has been deregistered. **Action**: No action is required.
08	**Equate Symbol**: IFAEDDRG_NOTAVAILABLE **Meaning**: Environmental error: The Deregister service is not available on this system. **Action**: Avoid calling the Deregister service on this system.
12	**Equate Symbol**: IFAEDDRG_NOTREGISTERED **Meaning**: User error: The product identified by the Prodtoken parameter was not registered. **Action**: In Prodtoken. provide a correct product token, as returned by the Register service.
16	**Equate Symbol**: IFAEDDRG_NOTTASKMODE **Meaning**: User error: The service was not called in task mode. **Action**: Avoid calling in this environment.
20	**Equate Symbol**: IFAEDDRG_XM **Meaning**: User error: The service was called in cross-memory mode but requires PASN=HASN=SASN. **Action**: Avoid calling in this environment.
24	**Equate Symbol**: IFAEDDRG_NOTAUTH **Meaning**: User error: A caller running in problem state tried to deregister a product that had been registered by an authorized caller (a program running in supervisor state or under a system key). **Action**: Avoid trying to deregister a product registered by an authorized caller.
36	**Equate Symbol**: IFAEDDRG_LOCKED **Meaning**: User error: The service was called while holding a system lock. **Action**: Avoid calling in this environment.
40	**Equate Symbol**: IFAEDDRG_FRR **Meaning**: User error: The service was called while having a functional recovery routine (FRR) established. **Action**: Avoid calling in this environment.

Query_Status Service (IFAEDSTA)

Use the Query_Status service (IFAEDSTA) to request information about the registration or enablement status of a particular product. The system will indicate, through a combination of return code value and output area content:
- If the product is registered (running)

Query_Status Service (IFAEDSTA)

- If the product is enabled

When it searches for the product you identify, the system does not use wildcard matching; there is no special treatment for a wildcard character (* or ?). You can, however, indicate fields that are not important to your search, and the system will try to find the best match it can for the parameters that you provide. If two matches are equivalently good, and one of them contains a registration from the current home address space, then that match is used.

Syntax

```
CALL IFAEDSTA
                        (Prodowner
                        ,Prodname
                        ,Featurename
                        ,ProdID
                        ,Outputinfo
                        ,Featureslen
                        ,Features
                        ,Returncode)
```

In C: the syntax is similar. You can use either of the following techniques to invoke the service:

1. `ifaedsta (Type,...Returncode);`
   When you use this technique, you must link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB.

2. `ifaedsta_byaddr (Type,...Returncode);`
   This second technique requires AMODE=31, and, before you issue the CALL, you must verify that the IFAEDSTA service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

In Assembler: Link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB unless you use either of the following techniques as an alternative to CALL IFAEDSTA:

1.
```
   LOAD EP=IFAEDSTA
   Save the entry point address
   ...
   Put the saved entry point address into R15
   Issue CALL (15),...
```

2.
```
   L   15,X'10'                 Get CVT
   L   15,X'8C'(,15)            Get ECVT
   L   15,X'1C0'(,15)
   L   15,4(,15)
   L   15,8(,15)                Get address of IFAEDSTA
   CALL (15),(...)
```
   Both of these techniques require AMODE=31. If you use the second technique, before you issue the CALL, you must verify that the IFAEDDRG service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

Note: This service is not available in Java.

Parameters

,Prodowner

Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Specifies the name of the product owner you are searching for. IBM products always use IBM CORP or IBM_CORP. If the product owner is not important to your search, set the first character of the field to an EBCDIC blank or hexadecimal zeroes.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Prodname

Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Specifies the name of the product you are searching for. If the product name is not important to your search, set the first character of the field to EBCDIC blank or hexadecimal zeroes.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Featurename

Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Specifies the name of the feature you are searching for. If the feature name is not important to your search, set the first character of the field to EBCDIC blank or hexadecimal zeroes.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,ProdID

Supplied parameter:
- Type: EBCDIC
- Length: 8 bytes

ProdID specifies the product identifier you are searching for. IBM products use the product's program number as the product identifier. If the product identifier is not important to your search, set the first character of the field to EBCDIC blank or hexadecimal zeroes.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), and period (.). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the name is less than 8 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Outputinfo
Returned parameter:
- Type: Character
- Length: 16

Specifies an output area, mapped by DSECT EDOI (in mapping macro IFAEDIDF) or structure EDOI (in C include file IFAEDC). If the return code is 0, this area contains information about the product you defined.

,Featureslen
Supplied parameter:
- Type: Integer
- Range: 0-1024
- Length: Full word

Featureslen specifies the length of the Features parameter that follows.

,Features
Returned parameter:
- Type: Character (EBCDIC recommended)
- Length: 1-1024 bytes

Features contains information provided by the caller of the Register service, and you need documentation from that caller about the length, format, and use of the information.

If the information is larger than the length you specify in Featureslen, the system returns only the information that fits in the area you provide. In that case, bit EdoiNotAllFeaturesReturned and field EdoiNeededFeaturesLen are set in the outputinfo area. You can use the length to call the Query_Status service again with an expanded area.

If you are not expecting any information from the caller of the Register service, code 0 in the Featureslen parameter. This system will then ignore the Features parameter, but the service syntax requires that you supply a value.

,Returncode
Returned parameter:
- Type: Integer
- Length: Full word

Returncode contains the return code from the Query_Status service.

Return Codes

When the Query_Status service returns control to the caller, Returncode contains the return code. To obtain the equates for the return codes:

- If you are coding in assembler, include mapping macro IFAEDIDF, described in *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*.
- If you are coding in C, use the include file IFAEDC. See "IFAEDC" on page 2-21.

The following table describes the return codes, shown in decimal.

Return Code (decimal)	Equate Symbol Meaning and Action
00	**Equate Symbol**: IFAEDSTA_SUCCESS **Meaning**: The product/feature combination is known to be registered or to be enabled or disabled. **Action**: Check the outputinfo area for further information.
04	**Equate Symbol**: IFAEDSTA_NOTDEFINED **Meaning**: The product/feature combination is **not** known to be registered or to be enabled or disabled. **Action**: Check that the operands are correct.
08	**Equate Symbol**: IFAEDSTA_NOTAVAILABLE **Meaning**: Environmental error: The Status service is not available on this system. **Action**: Avoid calling the Status service on this system.
16	**Equate Symbol**: IFAEDSTA_NOTTASKMODE **Meaning**: User error: The service was not called in task mode. **Action**: Avoid calling in this environment.
20	**Equate Symbol**: IFAEDSTA_XM **Meaning**: User error: The service was called in cross-memory mode but requires HASN=PASN=SASN. **Action**: Avoid calling in this environment.
36	**Equate Symbol**: IFAEDSTA_LOCKED **Meaning**: User error: The service was called while holding a system lock. **Action**: Avoid calling in this environment.
40	**Equate Symbol**: IFAEDSTA_FRR **Meaning**: User error: The service was called while having a functional recovery routine (FRR) established. **Action**: Avoid calling in this environment.

List_Status Service (IFAEDLIS)

Use the List_Status service (IFAEDLIS) to request information about the registration and enablement of one or more products. The system returns information about the products that match the product definition you supply.

You can also use the List_Status service to determine what, according to the current policy, the enablement state would be for the product you define. You might use this service to determine whether or not registering the product would require a change to the enablement policy in IFAPRDxx.

The system returns the information in the answer area you specify on the List_Status request:

List_Status Service (IFAEDLIS)

- In assembler language, the answer area is mapped by DSECTs EDAAHDR and EDAAE in mapping macro IFAEDIDF.
- In C language, the answer area is mapped by structures EDAAHDR and EDAAE in include file IFAEDC.

EDAAHDR maps information about the request, including the number of entries returned. EDAAE maps each returned entry.

Syntax

```
CALL IFAEDLIS,              (Type
                           ,Prodowner
                           ,Prodname
                           ,Featurename
                           ,ProdID
                           ,Anslen
                           ,Ansarea
                           ,Returncode)
```

In C: the syntax is similar. You can use either of the following techniques to invoke the service:

1. `ifaedlis (Type,...Returncode);`
   When you use this technique, you must link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB.

2. `ifaedlis_byaddr (Type,...Returncode);`
   This second technique requires AMODE=31, and, before you issue the CALL, you must verify that the IFAEDLIS service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

In Assembler: Link edit your program with a linkage-assist routine (also called a stub) in SYS1.CSSLIB unless you use either of the following techniques as an alternative to CALL IFAEDLIS:

1. ```
   LOAD EP=IFAEDLIS
   Save the entry point address
   ...
   Put the saved entry point address into R15
   Issue CALL (15),...
   ```

2. ```
   L   15,X'10'                 Get CVT
   L   15,X'8C'(,15)            Get ECVT
   L   15,X'1C0'(,15)
   L   15,4(,15)
   L   15,12(,15)                Get address of IFAEDLIS
   CALL (15),(...)
   ```
   Both of these techniques require AMODE=31. If you use the second technique, before you issue the CALL, you must verify that the IFAEDDRG service is available (in the CVT, both CVTOSEXT and CVTOS390 bits are set on).

Note: This service is not available in Java.

Parameters

Type
   Supplied parameter:

- Type: Integer
- Length: Full word

Identifies the type of list request. The field must contain a value that represents a combination of one or more of the possible types. You add the values to create the full word. Do not specify a type more than once. The possible types, and their meanings, are:

Ifaedlis_Type_Registered
> The system is to return data about any matching products that are registered. The number of entries returned appears in field EdaahNumR in the answer area. The address of the first entry appears in field EdaahFirstRAddr. DSECT EDAAE maps each entry. If you specify * or ? in the product definition, the system treats the character as a wildcard character.

Ifaedlis_Type_State
> The system is to return data about the current policy state (enabled or disabled) of any matching products. The number of entries returned appears in field EdaahNumS. The address of the first entry appears in field EdaahFirstSAddr. DSECT EDAAE maps each entry. If you specify * or ? in the product definition, the system treats the character as a wildcard character.

Ifaedlis_Type_Status
> The system is to return data about the enablement policy entry that would apply if the specified product registered. If there is no matching entry, the system sets Field EdaahStatusAddr in the answer area to 0; otherwise, it contains the address of the entry (mapped by DSECT EDAAE).
>
> **Note:** For this request type, the system does not use wildcard matching when it searches the policy. If you specify * or ? in the product definition, the system does not treat the character as a wildcard character. To indicate that a field is not important, however, you can set the first character of the field to an EBCDIC blank or hexadecimal zeroes.

Ifaedlis_Type_NoReport
> Specify this request type to indicate the system is to return information about all matching entries, including those that registered with **Ifaedreg_Type_NoReport**.

,Prodowner
> Supplied parameter:
> - Type: EBCDIC
> - Length: 16 bytes

Specifies the name of the product owner you are searching for. IBM products always use IBM CORP or IBM_CORP.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), period (.), asterisk (*), or question mark (?). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the request specifies **Ifaedlis_Type_Registered** or **Ifaedlis _Type_State**, the system treats * and ? as wildcard characters; it uses wildcard matching. When you specify **Ifaedlis_Type_Status**, the system does not use wildcard matching,

and * or ? receive no special treatment. If the product owner is not important to your search, set the first character of the field to an EBCDIC blank or hexadecimal zeroes.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Prodname
Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Specifies the name of the product you are searching for.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), period (.), asterisk (*), or question mark (?). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the request specifies **Ifaedlis_Type_Registered** or **Ifaedlis _Type_State**, the system treats * and ? as wildcard characters; it uses wildcard matching. When you specify **Ifaedlis_Type_Status**, the system does not use wildcard matching, and * or ? receive no special treatment. If the product name is not important to your search, set the first character of the field to an EBCDIC blank or hexadecimal zeroes.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Featurename
Supplied parameter:
- Type: EBCDIC
- Length: 16 bytes

Specifies the name of the feature you are searching for.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), period (.), asterisk (*), or question mark (?). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the request specifies **Ifaedlis_Type_Registered** or **Ifaedlis _Type_State**, the system treats * and ? as wildcard characters; it uses wildcard matching. When you specify **Ifaedlis_Type_Status**, the system does not use wildcard matching, and * or ? receive no special treatment. If the feature name is not important to your search, set the first character of the field to an EBCDIC blank or hexadecimal zeroes.

If the name is less than 16 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,ProdID
Supplied parameter:
- Type: EBCDIC
- Length: 8 bytes

ProdID specifies the product identifier you are searching for. IBM products, for example, use the product's program number as the product identifier.

The characters can be upper-case or lower-case alphabetics, numerics, national (@, #, $), underscore (_), slash (/), hyphen (-), period (.), asterisk (*), or question mark (?). You can use embedded blanks.

The system translates underscores to blanks for comparison and display, and it performs all comparisons in upper case.

If the request specifies **Ifaedlis_Type_Registered** or **Ifaedlis _Type_State**, the system treats * and ? as wildcard characters; it uses wildcard matching. When you specify **Ifaedlis_Type_Status**, the system does not use wildcard matching, and * or ? receive no special treatment. If the product identifier is not important to your search, set the first character of the field to an EBCDIC blank or hexadecimal zeroes.

If the name is less than 8 bytes, left-justify the name in the field and pad it on the right with EBCDIC blanks.

,Anslen
Supplied parameter:
- Type: Integer
- Minimum Value: 32
- Length: Full word

Specifies the length of the answer area parameter that follows. Specify a value of at least 32, the length of the answer area header (DSECT EDAAHDR in macro IFAEDIDF) that the system returns. Add 72 for each entry that you expect the system to return.

,Ansarea
Returned parameter:
- Type: Character
- Length: Specified on **Anslen** parameter

The answer area where the system is to place information about the request and the entries that match the product definition. The contents depend on the type of the request:

- If you specified **Ifaedlis_Type_Registered**, the answer area consists of a header area and a queue of 0 or more entries. The number of entries is in EDAAHNUMR, and EDAAHFIRSTRADDR points to the first entry. If you did not specify **Ifaedlis_Type_Registered**, both fields are 0.
- If you specified **Ifaedlis_Type_State**, the answer area consists of a header area and a queue of 0 or more entries. The number of entries is in EDAAHNUMS, and EDAAHFIRSTSADDR points to the first entry. If you did not specify **Ifaedlis_Type_State**, both fields are 0.
- If you specified **Ifaedlis_Type_Status**, the answer area consists of a header area and a single entry. EDAAHSTATUSADDR points to an entry that defines the policy that would be used to determine whether the product is enabled or disabled. The field is 0 if there is no matching policy entry, and it is always 0 when you did not specify **Ifaedlis_Type_Status**.

If the returned information exceeds the length you specify in Anslen, the system returns only the information that fits in the area you provided. EDAAHTLEN indicates the total length of the information available to be returned. If the length is longer than the length you specified in Anslen, increase Anslen and issue the request again.

,Returncode
Returned parameter:
- Type: Integer

List_Status Service (IFAEDLIS)

 • Length: Full word

Returncode contains the return code from the List_Status service.

Return Codes

When the List_Status service returns control to the caller, Returncode contains the return code. To obtain the equates for the return codes:

 • If you are coding in assembler, include mapping macro IFAEDIDF, described in *z/OS MVS Data Areas, Vol 2 (DCCB-ITZYRETC)*.

 • If you are coding in C, use the include file IFAEDC. See "IFAEDC" on page 2-21.

The following table describes the return codes, shown in decimal.

Return Code (decimal)	Equate Symbol Meaning and Action
00	**Equate Symbol**: IFAEDLIS_SUCCESS **Meaning**: The system returned all the requested data. **Action**: No action is required.
04	**Equate Symbol**: IFAEDLIS_NOTALLDATARETURNED **Meaning**: The answer area was too small. Some of the requested data was not returned. **Action**: Provide a larger answer area and call the service again.
08	**Equate Symbol**: IFAEDLIS_NOTAVAILABLE **Meaning**: Environmental error: The IFAEDLIS service is not available on this system. **Action**: Avoid calling the IFAEDLIS service on this system.
12	**Equate Symbol**: IFAEDLIS_ANSAREATOOSMALL **Meaning**: User error: The answer area length you provided was less than the minimum needed, 32. **Action**: Provide a larger answer area.
16	**Equate Symbol**: IFAEDSTA_NOTTASKMODE **Meaning**: User error: The service was not called in task mode. **Action**: Avoid calling in this environment.
20	**Equate Symbol**: IFAEDLIS_XM **Meaning**: User error: The service was called in cross-memory mode but requires PASN=HASN=SASN. **Action**: Avoid calling in this environment.
32	**Equate Symbol**: IFAEDLIS_BADTYPE **Meaning**: User error: The type parameter did not specify a word formed from any combination of Ifaedlis_Type_Registered, Ifaedlis_Type_State, Ifaedlis_Type_Status, and Ifaedlis_Type_Noreport. **Action**: Correct the parameter.
36	**Equate Symbol**: IFAEDLIS_LOCKED **Meaning**: User error: The service was called while holding a system lock. **Action**: Avoid calling in this environment.

Return Code (decimal)	Equate Symbol Meaning and Action
40	**Equate Symbol**: IFAEDLIS_FRR **Meaning**: User error: The service was called while having a functional recovery routine (FRR) established. **Action**: Avoid calling in this environment.

IFAEDC

For the C programmer, include file IFAEDC provides equates for return codes and data constants, such as Register service request types. To use IFAEDC, copy the file from SYS1.SAMPLIB to the appropriate local C library. The contents of the file are displayed in Figure 2-1 on page 2-22.

```
#ifndef __IFAED

#define __IFAED

/*********************************************************************
 *                                                                   *
 *  Name: IFAEDC                                                     *
 *                                                                   *
 *  Descriptive Name: SMF Product enable/disable services C declares *
 *                                                                   */
/*01* PROPRIETARY STATEMENT=                                         */
/***PROPRIETARY_STATEMENT********************************************/
/*                                                                  */
/*                                                                  */
/* LICENSED MATERIALS - PROPERTY OF IBM                             */
/* THIS MACRO IS "RESTRICTED MATERIALS OF IBM"                      */
/* 5645-001 (C) COPYRIGHT IBM CORP. 1996                            */
/* SEE COPYRIGHT INSTRUCTIONS                                       */
/*                                                                  */
/* STATUS= HBB6601                                                  */
/*                                                                  */
/***END_OF_PROPRIETARY_STATEMENT************************************/
/*                                                                  */
/*01* EXTERNAL CLASSIFICATION: GUPI                                 */
/*01* END OF EXTERNAL CLASSIFICATION:                              */
/*                                                                  */
/*  Function:                                                        *
 *    IFAEDC defines types, related constants, and function         *
 *    prototypes for the use of SMF Product enable/disable services *
 *    from the C language                                           *
 *                                                                  *
 *  Usage:                                                          *
 *    #include <IFAEDC.H>                                           *
 *                                                                  *
 *  Notes:                                                          *
 *  1. This member should be copied from SAMPLIB to the             *
 *     appropriate local C library.                                 *
 *                                                                  *
 *  2. The Product enable/disable services do not use a null        *
 *     character to terminate strings. The services expect the      *
 *     character operands to be a fixed-length type.                *
 *     Use memcpy to move into and from these fields.               *
 *                                                                  *
 *  Change Activity:                                                *
 *$L0=PRDEDSMF,HBB6601, 950601, PDXB: SMF Product enable/disable    *
 *                                                                  *
 *********************************************************************/
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 1 of 10)

```
/*******************************************************************
 *          Type Definitions for User Specified Parameters         *
 *******************************************************************/

/*  Type for TYPE operand of IFAEDREG                              */
typedef int  IfaedType;

/*  Type for Product Owner                                         */
typedef char IfaedProdOwner??(16??);

/*  Type for Product Name                                          */
typedef char IfaedProdName??(16??);

/*  Type for Feature Name                                          */
typedef char IfaedFeatureName??(16??);

/*  Type for Product Version                                       */
typedef char IfaedProdVers??(2??);

/*  Type for Product Release                                       */
typedef char IfaedProdRel??(2??);

/*  Type for Product Modification level                            */
typedef char IfaedProdMod??(2??);

/*  Type for Product ID                                            */
typedef char IfaedProdID??(8??);

/*  Type for Product Token                                         */
typedef char IfaedProdToken??(8??);

/*  Type for Features Length                                       */
typedef int  IfaedFeaturesLen;

/*  Type for Return Code                                           */
typedef int  IfaedReturnCode;
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 2 of 10)

```
/*********************************************************************
 *          Type Definitions for User Specified Parameters          *
 *********************************************************************/

/* Type for user supplied EDOI                                     */
typedef struct ??<
  struct ??<
    int EdoiRegistered : 1;          /* The product is registered  */
    int EdoiStatusNotDefined : 1;    /* The product is not known to
                          be enabled or disabled                   */
    int EdoiStatusEnabled    : 1;    /* The product is enabled      */
    int EdoiNotAllFeaturesReturned : 1;   /* The featureslen
                          area was too small to hold the features
                          provided at registration time. Field
                          EdoiNeededFeaturesLen contains the size
                          provided at registration time.           */
    int Rsvd0 : 4;                        /* Reserved              */
        ??> EdoiFlags ;
  char Rsvd1??(3??);           /* Reserved                         */
  int EdoiNeededFeaturesLen;  /* The featureslen size provided at
                                    registration time              */
  struct ??<
    IfaedProdVers EdoiProdVers;      /* The version information
                    provided at registration time                 */
    IfaedProdRel  EdoiProdRel;       /* The release information
                    provided at registration time                 */
    IfaedProdMod  EdoiProdMod;       /* The mod level information
                    provided at registration time                 */
        ??> EdoiProdVersRelMod;
  char Rsvd2??(2??);           /* Reserved                         */
??> EDOI;

/* Type for user supplied EDAAHDR                                  */
typedef struct ??<
  int EdaahNumR;                        /* Number of EDAAE entries which
                    follow indicating registered entries. The first one
                    is pointed to by EdaahFirstRAddr.             */
  int EdaahNumS;                        /* Number of EDAAE entries which
                    follow indicating state entries. The first one
                    is pointed to by EdaahFirstSAddr.             */
  int EdaahTLen;                        /* Total length of answer area
                    needed to contain all the requested information.
                    This includes the area for the records
                    that were returned on this call.              */
  void *EdaahFirstRAddr;                /* Address of first registered
                    entry EDAAE                                   */
  void *EdaahFirstSAddr;                /* Address of first state entry
                    EDAAE                                         */
  void *EdaahStatusAddr;                /* Address of the EDAAE that
                    represents the policy entry that would be used to
                    determine if the input product was enabled. 0 if
                    no such policy entry exists.                  */
  char Rsvd1??(8??);                    /*  Reserved               */
??> EDAAHDR;
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 3 of 10)

```
/*  Type for user supplied EDAAE                                  */
typedef struct ??<
  void *EdaaeNextAddr;             /* Address of next EDAAE. EdaahNumR
                   (for the registered queue) or EdaahNumS (for the
                   state queue) must be used to determine how far
                   along this chain to go. Not relevant for
                   EdaahStatusAddr.                              */
  struct ??<
    IfaedProdOwner   EdaaeProdOwner; /* Product owner            */
    IfaedProdName    EdaaeProdName;  /* Product name             */
    IfaedFeatureName EdaaeFeatureName; /* Feature name           */
    IfaedProdVers    EdaaeProdVers;  /* Product version          */
    IfaedProdRel     EdaaeProdRel;   /* Product release          */
    IfaedProdMod     EdaaeProdMod;   /* Product mod level        */
    IfaedProdID      EdaaeProdID;    /* Product ID               */
        ??> EdaaeInfo;
  struct ??<
    int EdaaeStatusNotDefined : 1;   /* This will never be on for
                   entries on the state queue. If on, indicates that
                   the state information does not have an entry that
                   matches this product.                        */
    int EdaaeStatusEnabled   : 1;    /* If on, indicates that the
                   product is considered to be enabled          */
    int EdaaeNoReport  : 1;          /* This will never be on for
                   entries on the state queue. If on, indicates that
                   the product registered with
                   Ifaedreg_Type_NoReport.                      */
    int EdaaeLicensedUnderProd : 1;  /* This will never be on for
                   entries on the state queue. If on, indicates that
                   the product registered with
                   Ifaedreg_Type_LicensedUnderProd.             */
    int Rsvd0 : 4;                   /* Reserved                 */
        ??> EdaaeFlags ;
  char Rsvd1??(1??);           /* Reserved                       */
  int EdaaeNumInstances;       /* Number of concurrent instances of
                                  this registration.            */
??> EDAAE;
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 4 of 10)

```
/********************************************************************
 *         Fixed Service Parameter and Return Code Defines         *
 ********************************************************************/

/*  Product enable/disable Register Constants                     */
#define Ifaedreg_Type_Standard          0
#define Ifaedreg_Type_Required          2
#define Ifaedreg_Type_NoReport          4
#define Ifaedreg_Type_LicensedUnderProd 8
#define Ifaedreg_Type_DisabledMessage   16
#define Ifaedreg_Type_NotFoundDisabled  32
#define IFAEDREG_TYPE_STANDARD          0
#define IFAEDREG_TYPE_REQUIRED          2
#define IFAEDREG_TYPE_NOREPORT          4
#define IFAEDREG_TYPE_LICENSEDUNDERPROD 8
#define IFAEDREG_TYPE_DISABLEDMESSAGE   16
#define IFAEDREG_TYPE_NOTFOUNDDISABLED  32

/*  Product enable/disable Register Return codes                  */
#define Ifaedreg_Success                0
#define Ifaedreg_Disabled               4
#define Ifaedreg_NotAvailable           8
#define Ifaedreg_LimitExceeded          12
#define Ifaedreg_NotTaskMode            16
#define Ifaedreg_XM                     20
#define Ifaedreg_BadFeaturesLen         24
#define Ifaedreg_NoStorage              28
#define Ifaedreg_BadType                32
#define Ifaedreg_Locked                 36
#define Ifaedreg_FRR                    40
#define IFAEDREG_SUCCESS                0
#define IFAEDREG_DISABLED               4
#define IFAEDREG_NOTAVAILABLE           8
#define IFAEDREG_LIMITEXCEEDED          12
#define IFAEDREG_NOTTASKMODE            16
#define IFAEDREG_XM                     20
#define IFAEDREG_BADFEATURESLEN         24
#define IFAEDREG_NOSTORAGE              28
#define IFAEDREG_BADTYPE                32
#define IFAEDREG_LOCKED                 36
#define IFAEDREG_FRR                    40
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 5 of 10)

```
/*  Product enable/disable Deregister Return codes                      */
#define Ifaeddrg_Success            0
#define Ifaeddrg_NotAvailable       8
#define Ifaeddrg_NotRegistered      12
#define Ifaeddrg_NotTaskMode        16
#define Ifaeddrg_XM                 20
#define Ifaeddrg_NotAuth            24
#define Ifaeddrg_Locked             36
#define Ifaeddrg_FRR                40
#define IFAEDDRG_SUCCESS            0
#define IFAEDDRG_NOTAVAILABLE       8
#define IFAEDDRG_NOTREGISTERED      12
#define IFAEDDRG_NOTTASKMODE        16
#define IFAEDDRG_XM                 20
#define IFAEDDRG_NOTAUTH            24
#define IFAEDDRG_LOCKED             36
#define IFAEDDRG_FRR                40

/*  Product enable/disable Status Return codes                          */
#define Ifaedsta_Success            0
#define Ifaedsta_NotDefined         4
#define Ifaedsta_NotAvailable       8
#define Ifaedsta_NotTaskMode        16
#define Ifaedsta_XM                 20
#define Ifaedsta_Locked             36
#define Ifaedsta_FRR                40
#define IFAEDSTA_SUCCESS            0
#define IFAEDSTA_NOTDEFINED         4
#define IFAEDSTA_NOTAVAILABLE       8
#define IFAEDSTA_NOTTASKMODE        16
#define IFAEDSTA_XM                 20
#define IFAEDSTA_LOCKED             36
#define IFAEDSTA_FRR                40
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 6 of 10)

```
/*  Product enable/disable List Constants                      */
#define Ifaedlis_Type_Registered      1
#define Ifaedlis_Type_State           2
#define Ifaedlis_Type_Status          4
#define Ifaedlis_Type_NoReport        8
#define IFAEDLIS_TYPE_REGISTERED       1
#define IFAEDLIS_TYPE_STATE           2
#define IFAEDLIS_TYPE_STATUS          4
#define IFAEDLIS_TYPE_NOREPORT        8


/*  Product enable/disable List Return codes                   */
#define Ifaedlis_Success              0
#define Ifaedlis_NotAllDataReturned   4
#define Ifaedlis_NotAvailable         8
#define Ifaedlis_AnsAreaTooSmall      12
#define Ifaedlis_NotTaskMode          16
#define Ifaedlis_XM                   20
#define Ifaedlis_BadType              32
#define Ifaedlis_Locked               36
#define Ifaedlis_FRR                  40
#define IFAEDLIS_SUCCESS              0
#define IFAEDLIS_NOTALLDATARETURNED   4
#define IFAEDLIS_NOTAVAILABLE         8
#define IFAEDLIS_ANSAREATOOSMALL      12
#define IFAEDLIS_NOTTASKMODE          16
#define IFAEDLIS_XM                   20
#define IFAEDLIS_BADTYPE              32
#define IFAEDLIS_LOCKED               36
#define IFAEDLIS_FRR                  40
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 7 of 10)

```
/********************************************************************
 *           Function Prototypes for Service Routines             *
 ********************************************************************/

#ifdef __cplusplus
   extern "OS" ??<
#else
  #pragma linkage(ifaedreg_calltype,OS)
  #pragma linkage(ifaeddrg_calltype,OS)
  #pragma linkage(ifaedsta_calltype,OS)
  #pragma linkage(ifaedlis_calltype,OS)
#endif
typedef void ifaedreg_calltype(
   IfaedType        __TYPE,       /* Input  - request type          */
   IfaedProdOwner   __PRODOWNER,  /* Input  - product owner         */
   IfaedProdName    __PRODNAME,   /* Input  - product name          */
   IfaedFeatureName __FEATURENAME,/* Input - feature name           */
   IfaedProdVers    __PRODVERS,   /* Input - product version        */
   IfaedProdRel     __PRODREL,    /* Input - product release        */
   IfaedProdMod     __PRODMOD,    /* Input - product modification   */
   IfaedProdID      __PRODID,     /* Input - product ID             */
   IfaedFeaturesLen __FEATURESLEN,/* Input - length of features     */
   void            *__FEATURES,   /* Input  - features area         */
   IfaedProdToken  *__PRODTOKEN,  /* Output - product token         */
   IfaedReturnCode *__RC);        /* Output - return code           */

typedef void ifaeddrg_calltype(
   IfaedProdToken   __PRODTOKEN,  /* Input - product token          */
   IfaedReturnCode *__RC);        /* Output - return code           */

typedef void ifaedsta_calltype(
   IfaedProdOwner   __PRODOWNER,  /* Input  - product owner         */
   IfaedProdName    __PRODNAME,   /* Input  - product name          */
   IfaedFeatureName __FEATURENAME,/* Input - feature name           */
   IfaedProdID      __PRODID,     /* Input - product ID             */
   EDOI            *__EDOI,       /* Output - output information     */
   IfaedFeaturesLen __FEATURESLEN,/* Input - length of features     */
   void            *__FEATURES,   /* Output - features area         */
   IfaedReturnCode *__RC);        /* Output - return code           */

typedef void ifaedlis_calltype(
   IfaedType        __TYPE,       /* Input  - request type          */
   IfaedProdOwner   __PRODOWNER,  /* Input  - product owner         */
   IfaedProdName    __PRODNAME,   /* Input  - product name          */
   IfaedFeatureName __FEATURENAME,/* Input - feature name           */
   IfaedProdID      __PRODID,     /* Input - product ID             */
   int              __ANSLEN,     /* Input - length of answer area  */
   void            *__ANSAREA,    /* Output - answer area           */
   IfaedReturnCode *__RC);        /* Output - return code           */
extern ifaedreg_calltype ifaedreg;
extern ifaeddrg_calltype ifaeddrg;
extern ifaedsta_calltype ifaedsta;
extern ifaedlis_calltype ifaedlis;
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 8 of 10)

```
#ifdef __cplusplus
   ??>
#endif


struct IFAED_PREDVT ??<
   ifaedreg_calltype* ifaed_regaddr;
   ifaeddrg_calltype* ifaed_drgaddr;
   ifaedsta_calltype* ifaed_staaddr;
   ifaedlis_calltype* ifaed_lisaddr;
??>;

struct IFAED_PRED ??<
   unsigned char ifaed_pred_filler1 ??(4??);
   struct IFAED_PREDVT * ifaed_predvt;
??>;

#ifndef __cplusplus
#define ifaedreg_byaddr(Type, Owner, Name, Fname, Vers, Rel, Mod,   \
                 Id, Flen, Fptr , Tptr, Rcptr)                      \
??<                                                                 \
 struct IFAED_PSA* ifaed_pagezero = 0;                              \
   ifaed_pagezero->ifaed_cvt->ifaed_cvtecvt->ifaed_ecvtpred->       \
           ifaed_predvt->ifaed_regaddr                              \
           (Type,Owner,Name,Fname,Vers,Rel,Mod,Id,Flen,Fptr,       \
            Tptr,Rcptr);                                            \
??>;
#define ifaeddrg_byaddr(Token, Rcptr)                               \
??<                                                                 \
 struct IFAED_PSA* ifaed_pagezero = 0;                              \
   ifaed_pagezero->ifaed_cvt->ifaed_cvtecvt->ifaed_ecvtpred->       \
           ifaed_predvt->ifaed_drgaddr                              \
           (Token,Rcptr);                                           \
??>;
#define ifaedsta_byaddr(Owner, Name, Fname, Id, Eptr, Flen,        \
                      Fptr, Rcptr)                                  \
??<                                                                 \
 struct IFAED_PSA* ifaed_pagezero = 0;                              \
   ifaed_pagezero->ifaed_cvt->ifaed_cvtecvt->ifaed_ecvtpred->       \
           ifaed_predvt->ifaed_staaddr                              \
           (Owner,Name,Fname,Id,Eptr,Flen,Fptr,Rcptr);             \
??>;
#define ifaedlis_byaddr(Type, Owner, Name, Fname, Id, Alen,        \
                      Aptr, Rcptr)                                  \
??<                                                                 \
 struct IFAED_PSA* ifaed_pagezero = 0;                              \
   ifaed_pagezero->ifaed_cvt->ifaed_cvtecvt->ifaed_ecvtpred->       \
           ifaed_predvt->ifaed_lisaddr                              \
           (Type,Owner,Name,Fname,Id,Alen,Aptr,Rcptr);             \
??>;
#endif
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 9 of 10)

```
struct IFAED_ECVT ??<
   unsigned char ifaed_ecvt_filler1  ??(448??);
   struct IFAED_PRED * ifaed_ecvtpred; /*
                                    product enable/disable block    */
   unsigned char ifaed_ecvt_filler2  ??(24??);
   unsigned char ifaed_ecvtpseq ??( 4??); /* product sequence number */
   IfaedProdOwner ifaed_ecvtpown;          /* product owner          */
   IfaedProdName ifaed_ecvtpnam;           /* product name           */
   IfaedProdVers ifaed_ecvtpver;           /* product version        */
   IfaedProdRel  ifaed_ecvtprel;           /* product release        */
   IfaedProdMod  ifaed_ecvtpmod;           /* product mod level       */
   unsigned char ifaed_ecvt_filler3  ??(26??);
??>;

struct IFAED_CVT ??<
   unsigned char ifaed_cvt_filler1  ??(116??);
  struct ??<
    int ifaed_cvtdcb_rsvd1 : 4;      /* Not needed               */
    int ifaed_cvtosext : 1;          /* If on, indicates that the
                 CVTOSLVL fields are valid                        */
    int ifaed_cvtdcb_rsvd2 : 3;      /* Not needed               */
        ??> ifaed_cvtdcb;
  unsigned char ifaed_cvt_filler2  ??(23??);
  struct IFAED_ECVT * ifaed_cvtecvt;
  unsigned char ifaed_cvt_filler3  ??(1120??);
  unsigned char ifaed_cvtoslv0;
  struct ??<
    int ifaed_cvtoslv1_rsvd1 : 6;    /* Not needed               */
    int ifaed_cvtprded : 1;          /* If on, indicates that the
                 product enable/disable services are available    */
    int ifaed_cvtoslv1_rsvd2 : 1;    /* Not needed               */
        ??> ifaed_cvtoslv1;
   unsigned char ifaed_cvt_filler4 ??(14??);
??>;


struct IFAED_PSA ??<
   char ifaed_psa_filler??(16??);
   struct IFAED_CVT* ifaed_cvt;
??>;

/*  End of SMF Product Enable/Disable Services Header              */

#endif
```

Figure 2-1. IFAEDC from SYS1.SAMPLIB (Part 10 of 10)

IFAEDC

Chapter 3. Examples

The following examples show possible uses of the registration services. The examples are written in assembler and Java.

Detailed information about the services appears in Chapter 2, "Coding Registration Services," on page 2-1

Registering a product, checking the status of another product, then deregistering the first product using assembler

Figure 3-1 shows code that registers a product, checks the status of another product, then deregisters the first product and uses the MF parameter of the CALL macro to generate reentrant code.

```
PUBEX1   CSECT
PUBEX1   AMODE 31
PUBEX1   RMODE ANY
         STM   14,12,12(13)
         LR    12,15
         USING PUBEX1,12
         GETMAIN RU,LV=DYNAREALEN
         LR    14,1
         ST    13,4(,14)
         ST    14,8(,13)
         LR    13,14
         USING DYNAREA,13
DYNAREA  DSECT
SAVEAREA DS    CL72
PUBEX1   CSECT
EXAMPLE1 DS    0H
**********************************************************************
* Register a product                                                *
**********************************************************************
         CALL  IFAEDREG,(RTYPE,ROWNER,RNAME,                         *
               RFEATURENAME,RVERSION,RRELEASE,                       *
               RMOD,RID,RFEATURESLEN,RFEATURES,PRODTOKEN,RETCODE),   *
               MF=(E,PL)
*
* Place code to check return code  here
*
**********************************************************************
* Check the status of another product                               *
**********************************************************************
         CALL  IFAEDSTA,(SOWNER,SNAME,SFEATURENAME,                  *
               SID,SOUTPUTINFO,                                      *
               SFEATURESLEN,SFEATURES,RETCODE),MF=(E,PL)
*
* Place code to check return code  here
*
```

Figure 3-1. Example 1 — Using IFAEDREG, IFAEDSTA and IFAEDDRG (Part 1 of 2)

```
      ********************************************************************
      * Deregister the product                                          *
      ********************************************************************
              CALL  IFAEDDRG,(PRODTOKEN,RETCODE),MF=(E,PL)
      *
      * Place code to check return code  here
      *
              B     ENDEXAMPLE
      *
      * Values for REGISTER
      *
      RTYPE     DC    AL4(IFAEDREG_TYPE_STANDARD)
      ROWNER    DC    CL16'VENDOR X'
      RNAME     DC    CL16'Y_PROD1 '
      RFEATURENAME DC CL16' '
      RID       DC    CL8'1234-567'
      RVERSION  DC    CL2'01'
      RRELEASE  DC    CL2'01'
      RMOD      DC    CL2'00'
      RFEATURESLEN DC AL4(L'RFEATURES)
      RFEATURES DC C'FEATURE1,FEATURE2OPT=2'
      *
      * Values for STATUS
      *
      SOWNER    DC    CL16'VENDOR Y'
      SNAME     DC    CL16'Y_PROD2 '
      SFEATURENAME DC CL16' '
      SID       DC    CL8'8888-888'
      SFEATURESLEN DC AL4(L'SFEATURES)
              IFAEDIDF ,            Return code information
      DYNAREA  DSECT
      RETCODE  DS    F
      PRODTOKEN DS   CL8
      SOUTPUTINFO DS CL16
      SFEATURES DS   CL1024
      PL       CALL  ,(,,,,,,,,,,,),MF=L  Call parm list for 12 parameters
      DYNAREALEN EQU *-DYNAREA
      PUBEX1   CSECT
      ENDEXAMPLE DS 0H
              LR    1,13            Dynamic area address
              L     13,4(,13)       Previous save area address
              FREEMAIN RU,A=(1),LV=DYNAREALEN
              LM    14,12,12(13)
              SLR   15,15
              BR    14
              END   PUBEX1
```

Figure 3-1. Example 1 — Using IFAEDREG, IFAEDSTA and IFAEDDRG (Part 2 of 2)

Obtaining a list of information about products that are registered using assembler

Figure 3-2 on page 3-3 shows code that obtains a list of information about products that are registered, including information about their enablement state and uses the MF parameter of the CALL macro to generate reentrant code..

```
PUBEX2   CSECT
PUBEX2   AMODE 31
PUBEX2   RMODE ANY
         STM   14,12,12(13)
         LR    12,15
         USING PUBEX2,12
         GETMAIN RU,LV=DYNAREALEN
         LR    14,1
         ST    13,4(,14)
         ST    14,8(,13)
         LR    13,14
         USING DYNAREA,13
DYNAREA  DSECT
SAVEAREA DS    CL72
PUBEX2   CSECT
EXAMPLE3 DS    0H
* Following is an assembler example of getting registration and
* state information about all of the products
         L     2,=AL4(INITEDAA)  Initial answer area size
         ST    2,SIZEEDAA       Save it
         GETMAIN RU,LV=(2)      Allocate the answer area
         ST    1,EDAA@          Save address of answer area
LAB1     DS    0H
         L     4,EDAA@          Address of answer area
         CALL  IFAEDLIS,(REQ_INFO,                              *
               ALL_OWNER,ALL_NAME,ALL_FN,ALL_ID,                *
               SIZEEDAA,(4),LRETCODE),MF=(E,PL)
         CLC   LRETCODE(4),=AL4(IFAEDLIS_NOTALLDATARETURNED) Warning?
         BNE   LAB2             No, request successful or error
*                               Yes, not enough room
         LR    3,2              Save current size
         L     2,EDAAHTLEN-EDAAHDR(4) Get required size
         FREEMAIN RU,A=(4),LV=(3) Release old area
         ST    2,SIZEEDAA       Save it
         GETMAIN RU,LV=(2)      Allocate new area
         ST    1,EDAA@          Save address of answer area
         B     LAB1             Retry List operation
LAB2     DS    0H
         CLC   LRETCODE(4),=AL4(IFAEDLIS_SUCCESS) Success?
         BNE   LAB3             No, error
```

Figure 3-2. Example 2 — Using IFAEDLIS (Part 1 of 3)

Examples

```
***********************************************************************
*                                                                     *
* Process information in answer area when RC=0                        *
*                                                                     *
***********************************************************************
         USING EDAAHDR,4        EDAAHDR DSECT
*
* Process registered entry information
*
         L     5,EDAAHNUMR       Find how many EDAAE registered entries
         LTR   5,5               Are there any entries
         BZ    LAB4              No, check state entries
         L     6,EDAAHFIRSTRADDR Get first entry
         USING EDAAE,6           EDAAE DSECT
LAB5     DS    0H                EDAAE loop
*
* Put code to process information contained in EDAAE here
*
         L     6,EDAAENEXTADDR Get next EDAAE
         BCT   5,LAB5            Continue while there are more
         DROP  6
*
* Process state entry information
*
LAB4     DS    0H                EDAAE loop
         L     5,EDAAHNUMS       Find how many EDAAE state entries
         LTR   5,5               Are there any entries
         BZ    LAB10             No, done
         L     6,EDAAHFIRSTSADDR Get first entry
         USING EDAAE,6           EDAAE DSECT
LAB6     DS    0H                EDAAE loop
*
* Put code to process information contained in EDAAE here
*
         L     6,EDAAENEXTADDR Get next EDAAE
         DROP  6
         BCT   5,LAB6            Continue while there are more
         B     LAB4              Skip error case
LAB3     DS    0H                Error return
*
*  Process error case
*
LAB10    DS    0H                Common path
         L     2,SIZEEDAA        Get size of area
         L     4,EDAA@           Get address of area
         FREEMAIN RU,A=(4),LV=(2) Release area
         B     ENDEXAMPLE
```

Figure 3-2. Example 2 — Using IFAEDLIS (Part 2 of 3)

```
          INITEDAA EQU   4096             Initial size of answer area
          DYNAREA  DSECT
          EDAA@    DS    A                Address of answer area
          SIZEEDAA DS    F                Size of answer area
          TEMPSIZE DS    F                Temporary
          LRETCODE DS    F                Return code
          PL       CALL  ,(,,,,,,,),MF=L  Call parameter list for 8 parameters
          PUBEX2   CSECT
          REQ_INFO DC    A(IFAEDLIS_TYPE_REGISTERED+IFAEDLIS_TYPE_STATE)
          ALL_OWNER DC   CL16'*'          Match all product owners
          ALL_NAME  DC   CL16'*'          Match all product names
          ALL_FN    DC   CL16'*'          Match all feature names
          ALL_ID    DC   CL8'*'           Match all product IDs
                   IFAEDIDF ,             Service equates
          DYNAREA  DSECT
          DYNAREALEN EQU *-DYNAREA
          PUBEX2   CSECT
          ENDEXAMPLE DS  0H
                   LR    1,13             Dynamic area address
                   L     13,4(,13)        Previous save area address
                   FREEMAIN RU,A=(1),LV=DYNAREALEN
                   LM    14,12,12(13)
                   SLR   15,15
                   BR    14
                   END   PUBEX2
```

Figure 3-2. Example 2 — Using IFAEDLIS (Part 3 of 3)

Registering and deregistering a product using Java

The IFAEDJReg class provides access to the z/OS product registration and deregistration services through Java. The IFAEDJReg class allows Java programs to use the product registration services by wrapping the system IFAEDREG (register) and IFAEDDRG (deregister) callable services.

A product is identified through various parameters such as product name, product owner, and feature name. These fields are set in the IFAEDJReg object and then the register method can be called.

After a successful registration, a registration token (also referred to as a product token) is returned by the system. This token is used by the deregister method to deregister the product.

The registration or deregistration return code provided by the system is returned by the methods.

The product registration Java support requires that the following Java level or higher be installed:

- IBM SDK for z/OS Java 2 Technology Edition, Version 1.4 PTF UQ93743, product number 5655-I56

To run an application that uses the product registration Java classes, you must add the jar file containing the product registration classes to your path, and add the native library to your libpath.

Examples

Note in the examples below, the default installation paths are shown. If you have changed the default by adding a path prefix, modify the commands accordingly.

Add the /usr/include/java_classes/ifaedjreg.jar file to your application classpath. In the z/OS UNIX System Services shell, this can be done with the command:

```
export CLASSPATH=/usr/include/java_classes/ifaedjreg.jar:$CLASSPATH
```

Add the path to the native library /usr/lib/java_runtime/libifaedjreg.so to your library path (LIBPATH). In the z/OS Unix System Services shell, this can be done with the command:

```
export LIBPATH=/usr/lib/java_runtime:$LIBPATH
```

The documentation for the using the methods in the Java classes is contained in the Javadoc for the IFAEDJReg class. The Java doc is installed to /usr/include/java_classes/ifadjregDoc.jar by default.

All of the Javadoc files for product registration have been included in the jar. To view the Javadoc, it is necessary to download the jar file in binary to your workstation, unjar the file to make the individual files accessible, and then use your browser to open the index.html file.

Example: Registering a product using Java

The product name ″TESTPROD″ with product owner ″IBM″ and product number 9999-999 is to be registered. If the product is disabled, the program will exit. Registration is done using the system service IFAEDREG.

```
IFAEDJReg reg = new IFAEDJReg();
reg.setRegisterType(IFAEDJReg.IFAEDREG_TYPE_STANDARD + IFAEDJReg.IFAEDREG_TYPE_NOTFOUNDDISABLED);
reg.setProductName("TESTPROD");
reg.setProductOwner("IBM");
reg.setProductID("9999-999");
int rc = reg.register(); // Invoke registration service
if (rc != IFAEDJReg.IFAEDREG_SUCCESS) {
  System.out.println("TESTPROD registration failed due to the return code from IFAEDJReg, rc="+rc);
  System.exit(1);
}
```

Example: Deregistering a product using Java

A previously registered product is to be deregistered. The same object that was used during registration is used for the deregistration. Deregistration is done using the system service IFAEDDRG.

```
nt rc = reg.deregister(); // Invoke deregistration service
if (rc != IFAEDJReg.IFAEDDRG_SUCCESS) {
  System.out.println("TESTPROD deregistration failed due to the return code from IFAEDJReg, rc="+rc);
  System.exit(2);
}
```

Appendix. Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features in z/OS enable users to:

- Use assistive technologies such as screen readers and screen magnifier software
- Operate specific or equivalent features using only the keyboard
- Customize display attributes such as color, contrast, and font size

Using assistive technologies

Assistive technology products, such as screen readers, function with the user interfaces found in z/OS. Consult the assistive technology documentation for specific information when using such products to access z/OS interfaces.

Keyboard navigation of the user interface

Users can access z/OS user interfaces using TSO/E or ISPF. Refer to *z/OS TSO/E Primer*, *z/OS TSO/E User's Guide*, and *z/OS ISPF User's Guide Vol I* for information about accessing TSO/E and ISPF interfaces. These guides describe how to use TSO/E and ISPF, including the use of keyboard shortcuts or function keys (PF keys). Each guide includes the default settings for the PF keys and explains how to modify their functions.

z/OS information

z/OS information is accessible using screen readers with the BookServer/Library Server versions of z/OS books in the Internet library at:

`www.ibm.com/servers/eserver/zseries/zos/bkserv/`

Notices

This information was developed for products and services offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

B-1

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Programming Interface Information

This book documents intended Programming Interfaces that allow the customer to write programs that obtain the services of z/OS.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:
* Hiperbatch
* IBM
* IBMLink
* MVS/ESA
* OS/390
* Resource Link
* z/OS

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Index

A
accessibility   A-1

C
checking product status   1-3
coding registration services   2-1

D
Deregister service   2-9
   example   3-1
   invoking   2-1
   Java example   3-5
   using   1-3
disability   A-1
dynamic enablement   1-1

E
enablement, product   1-1
examples, product registration services   3-1

I
IFAEDC include file   2-14, 2-16, 2-21
IFAEDDREG service
   Java example   3-5
IFAEDDRG service   2-9
   example   3-1
IFAEDIDF mapping macro   2-1, 2-14, 2-16
IFAEDLIS service   2-15
   example   3-2
IFAEDREG service   2-2
   example   3-1
   Java example   3-5
IFAEDSTA service   2-11
   example   3-1
IFAPRDxx   1-1, 2-3

J
Java
   deregistering a product   3-5
   registering a product   3-5

K
keyboard   A-1

L
List_Status service   2-15
   example   3-2
   invoking   2-1
   using   1-3

M
message retrieval tool, LookAt   vii

N
Notices   B-1

O
overview of product registration   1-1

P
product enablement   1-1
product registration
   checking product status   1-3
   coding the services   2-1
   Deregister service   2-9
   examples   3-1
   List_Status service   2-15
   overview   1-1
   Query_Status service   2-11
   Register service   2-2

Q
Query_Status service   2-11
   example   3-1
   invoking   2-1
   using   1-3

R
Register service   2-2
   example   3-1
   invoking   2-1
   Jave example   3-5
   using   1-2
registration services   1-1
registration, product
   See product registration
request type
   List_Status service   2-17
   Register service   2-4

S
shortcut keys   A-1

LookAt message retrieval tool   vii

Readers' Comments — We'd Like to Hear from You

z/OS
MVS Programming:
Product Registration

Publication No. SA22-7604-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you?　☐ Yes　☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Company or Organization

Phone No.

Address

IBM®

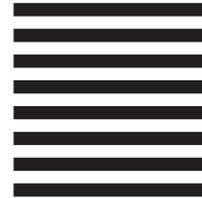Fold and Tape   **Please do not staple**   Fold and Tape

BUSINESS REPLY MAIL

FIRST-CLASS MAIL   PERMIT NO. 40   ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
2455 South Road
Poughkeepsie, NY
 12601-5400

Fold and Tape   **Please do not staple**   Fold and Tape

IBM ®

Program Number: 5694-A01

Printed in USA