VA FileMan 22.2; Patch DI*22.2*8

Data Access Control (DAC) User Guide

August 2017

Department of Veterans Affairs (VA)

Office of Information and Technology (OI&T)

Enterprise Program Management Office (EPMO)

Revision History

Date	Revision	Description	Author
08/07/2017	1.0	Initial Data Access Control (DAC) document released with VA FileMan Patch DI*22.2*8.	VA FileMan 22.2 Development Team

REF: For the current patch history related to this software, see the Patch Module (i.e., Patch User Menu [A1AE USER]) on FORUM.

Table of Contents

List of Figures

List of Tables

1 Introduction

VA FileMan has provided basic file access control since its inception. It allows Veterans Health Information Systems and Technology Architecture (VistA) applications to define letter-based codes that can be assigned to users to grant access to the data in a given file. This access could differ for various actions, such as:

- Reading data
- Writing data
- Access to the file's data dictionary

Kernel 8.0 released the File Access Security module. It expanded upon this concept via an explicit list in the NEW PERSON (#200) file of the files that each user may access and how.

Both of these methods permit or deny a user access to specific fields, or the file as a whole. Applications face an additional access control issue at the record level. Often users may view final or verified data but anything that is still considered "draft" is usually restricted via application options. Sensitive or "removed" (i.e., entered in error) data is often hidden. Actions available to take on a record can vary depending upon its current state or the credentials of an individual user.

VistA application developers have traditionally coded these behaviors, generally known as business rules, into their application options and remote procedure calls (RPCs). New Web clients are accessing data via services instead of RPCs. The Data Access Control (DAC) utility was developed to save those business rules in a place that can be called within a Web service. Applications can also use this tool directly within their own code for consistency. Potentially, VA FileMan application programming interfaces (APIs) could use this type of access control.

The Data Access Control utility, released with VA FileMan Patch DI*22.2*8, is based upon the XACML (eXtensible Access Control Markup Language) standard, an extension of XML. XACML assumes an attribute-based rule structure that can permit or deny a user access to a resource (e.g., a record in a file). It uses pre-defined target attribute values to match appropriate policies to the action being taken. Policies are made up of rules that can be combined as needed. This allows you to create simple or very complex access policies.

1.1 What is a Policy?

A policy is a set of rules that specify who may access a record, and how. Attribute name-value pairs, called **targets**, determine if the policy applies to a record; a conjunction can be used to indicate if all targets *must* match the record, or if only one match suffices. Each rule in a policy can have its own targets, to return different results based on different attribute values. Policies can return a message when a result is determined, and/or execute M code to perform a task, such as logging a user's access to the file.

1.2 Creating a Policy

When creating a policy you first identify the business rules that determine how to appropriately handle the data. For example, Lab wants to restrict who may view results before they are verified. In this example, a policy can be created for reading data from the LAB DATA (#63) file. This policy would have rules to permit or deny access based on the status of the result and the credentials of each user.

1.2.1 Create the Rules

A rule can be created for each possible scenario that *must* be considered. You need to define the following:

- Target—Criteria for applying the rule.
- Condition—Any conditions to be evaluated if the rule applies.
- Result—Result to return if the conditions are true.

1.2.1.1 Target

The target is a set of attributes that determine if the rule applies to a record. The name and desired value of each attribute is stored in a sub-file with the rule. A conjunction can be used to indicate if all targets *must* match, or if at least one suffices to apply the rule.

1.2.1.2 Condition

Rules can have additional conditions, which are simply Boolean expressions that are evaluated to determine the result of the rule, when it is applied. A condition points to an entry in the POLICY FUNCTION (#1.62) file, and can include a value for comparison. Conditions are particularly useful for evaluating requirements that are external to the record, such as user credentials. For example, VA FileMan provides a condition function to check if the user holds a specified security key. A conjunction can be used to indicate if all conditions *must* be true, or at least one, for the rule to be considered true.

REF: For a list of exported DI* functions, see "Appendix B—Exported DI* Functions."

1.2.1.3 Result

The result is the intended consequence or effect of the rule, if it applies and the conditions evaluate to true. Possible values are either:

- Permit
- Deny

Depending on the result of the rule, each rule can also:

- Return a message.
- Process an obligation function.

The function is M code stored as an entry in the POLICY FUNCTION (#1.62) file. It is executed based on the result and can perform tasks, such as logging access that has been granted or denied.

REF: For a list of exported DI* functions, see "Appendix B—Exported DI* Functions."

1.2.2 Create the Policy

Related rules can be collected together into a single policy, which can be evaluated to permit or deny access to a record. A policy can have its own set of target attributes, as well as messages and functions.

A policy should also include an **attribute function**. This is M code that is stored as an entry in the POLICY FUNCTION (#1.62) file, which extracts the attribute values from the record and stores them in the **DIVAL** array by subscripts that match the target names used in the policy and its rules.

Because a policy usually contains multiple rules, a **result function** *must* be defined as well. This is M code that is stored as an entry in the POLICY FUNCTION (#1.62) file, which based on the current value of the local variable **DIRESULT** sets the local variable **Y** to either of the following:

- **1—**True
- **0—**False

A policy can loop through rules until the following:

- **Permit** result occurs.
- **Deny** is encountered.
- Quits when any result is determined.

Some result functions can provide a default result if no policy or rule is found that applies to the record. VA FileMan provides five result functions that handle the most common combining algorithms.

REF: For more information on result functions, see the "Functions" section.

REF: For a list of exported DI* functions, see "Appendix B—Exported DI* Functions."

1.2.2.1 Functions

The POLICY FUNCTION (#1.62) file contains the following function types:

- **Attribute—**Written by the application developer to extract values from the record that are used to determine which policies and rules apply; it should populate the array DIVAL("attribute")="value", where 'attribute' matches the policy or rule targets.
- **Obligation—**Provided by the application developer to perform any needed tasks when a Permit or Deny result is obtained, such as logging access to a file. No input or output is required.
- **Condition—**Provides additional checks for rules that apply. VA FileMan exports a few common functions, but others can be provided by the developer as needed. A value can be specified to compare against, and is available for use in the function code as the variable **X**; the variable **Y** *must* be returned as:
  - **1—**True
  - **0—**False
- **Result—**Provided by VA FileMan; examine the variable DIRESULT to determine if a policy is satisfied. Like conditions, the variable **Y** is returned as either:
  - **1—**True
  - **0—**False

When function code is executed, the variable IEN holds the pointer to the POLICY (#1.6) file for the policy or rule being evaluated.

REF: For a list of exported DI* functions, see "Appendix B—Exported DI* Functions."

REF: For the full list of local variables that can be referenced, see "Appendix A—Key DI* Variables."

1.2.2.2 Application Actions

The APPLICATION ACTION (#1.61) file contains the actions that can be taken on a record in a given VistA file or sub-file. Each entry consists of a valid VistA file or sub-file number, and the free text name of an action that is supported by its application (e.g., read, cancel, sign, etc.).

A policy can be linked to an Application Action. When the policy evaluation API is invoked, the file number and action name passed in are used to find a match in this file; its associated policy is then evaluated to return a data access authorization result of Permit or Deny. More than one Application Action can be assigned the same policy.

NOTE: A policy *must* be linked to an Application Action to be evaluated for Data Access Control.

1.3 Distribution Files

The Data Access Control (DAC) software uses the **DIAC** namespace.

The DAC software distributes the following files:

Table 1: Data Access Control—Files

Global	File (#)	Description
^DIAC(1.6,	POLICY (#1.6)	This file is a self-referring, namespaced file, which is similar to the OPTION (#19) file. Rules are stored in a sub-file, much like menu items, and evaluated in sequence. If more complex policies are needed, policy sets can be created by grouping other policies or sets, drilling down the levels in sequence like a menu tree.
^DIAC(1.61,	APPLICATION ACTION (#1.61)	This file stores the list of actions that can be taken on a file or sub-file (e.g., read, cancel, sign, etc.). Each action can be mapped to a policy that is evaluated when that kind of access to data is requested.
^DIAC(1.62,	POLICY FUNCTION (#1.62)	Supporting M code for policies is implemented as M functions and stored as entries in this file.

The DAC software distributes the following options:

Table 2: Data Access Control—Options

Option Name	Option Text	Description
DIACCESS	Data Access Control	This menu contains options that allow creation and management of Data Access Control policies for VistA files. It includes the following option items (listed in display order): • Set Up Application Actions [DIAC ACTIONS] • Edit/Create an Action Policy [DIAC EDIT] • Test a Policy [DIAC TEST] • Disable a Policy [DIAC DISABLE] • Delete a Policy [DIAC DELETE] • Print Actions/Policies [DIAC PRINT] • Policy Functions [DIAC FUNCTIONS]
DIAC ACTIONS	Set Up Application Actions	This option provides access to the APPLICATION ACTION (#1.61) file, which allows the user to edit an existing or create a new Application Action.
DIAC DELETE	Delete a Policy	This option allows users to delete a DAC policy.
DIAC DISABLE	Disable a Policy	This option provides a way to quickly disable a DAC policy.
DIAC EDIT	Edit/Create an Action Policy	This option provides access to the DAC Policy Editor.
DIAC FUNCTIONS	Policy Functions	This option provides access to the POLICY FUNCTION (#1.62) file, which allows the user to edit an existing or create a new Policy Function.
DIAC PRINT	Print Actions/Policies	This option allows users to print a summary list of application actions and their associated policies, or

Option Name	Option Text	Description
		details of one policy.
DIAC TEST	Test a Policy	This option allows users to test a DAC policy.

The DAC software distributes the following protocols (actions):

Table 3: Data Access Control—Protocols (Actions)

Protocol Name	Protocol Text	Description
DIAC ACTION MENU	Policy Action Menu	This action menu contains options that allow creation and management of Data Access Control policies for VistA files. It includes the following action items (listed in sequence order): • Add/Remove Members [DIAC MEMBERS] • Edit a Policy [DIAC EDIT] • Delete a Policy [DIAC DELETE] • Blank Line [DIAC BLANK LINE] • Expand/Collapse [DIAC EXPAND] • Inquiry [DIAC INQUIRE] • Functions [DIAC FUNCTIONS] • Link to Appl Action [DIAC LINK ACTION] • Test Policy [DIAC TEST] • Select New Policy [DIAC CHANGE] • Quit [DIAC QUIT]
DIAC BLANK LIN	Blank Line	N/A
DIAC CHANGE	Select New Policy	This action allows the user to change the policy that is displayed in the list for editing.
DIAC DELETE	Delete a Policy	This action completely removes a rule or policy from the POLICY (#1.6) file, and cleans up all pointers to that item.
DIAC DISABLE	Disable a Policy	This action changes the value of the DISABLE (#.03) field in the POLICY (#1.6) file.
DIAC EDIT	Edit a Policy	This action invokes a ScreenMan form, which allows reviewing and editing of any rule or policy in the list.
DIAC EXPAND	Expand/Collapse	This action toggles the display to expanding a policy to see all of its rules or collapsing the members.
DIAC FUNCTIONS	Functions	This action reviews existing functions or creates new ones.
DIAC INQUIRE	Inquiry	This action displays a snapshot of the selected item in VA FileMan captioned format. It also lists all policies or sets to which the selected item is itself a member.
DIAC LINK ACTION	Link to Appl Action	This action assigns the policy to an Application Action.
DIAC MEMBERS	Add/Remove Members	This action creates a rule for the Preliminary (**P**) status.
DIAC QUIT	Quit	This action quits an action.
DIAC TEST	Test Policy	This action tests a policy. It has an optional trace

Protocol Name	Protocol Text	Description
		feature that shows the details of policy execution, including local variables.

The DAC software distributes the following routines:

Table 4: Data Access Control—Routines

Routine	Description
DIAC1	Policy Evaluation API.
DIAC1T	Test utility for Policies.
DIACLM	Policy Editor driver.
DIACLM1	Policy Editor actions.
DIACOPT	Data Access Control Options.
DIACP	Print Policy Reports.
DIACX	Policy utilities.

2 Policy Editor

The Data Access Control (DAC) utility provides a Policy Editor to facilitate creation and management of data access control policies. Policy members are indented in the list to show the membership hierarchy, with common actions and utilities available in the action menu below.

2.1 Policy

Figure 1 and Figure 2 are examples of creating a sample policy to view Chemistry results. Invoke the Policy Editor via the Data Access Control options on the VA FileMan Other Options menu, and create a new policy called LR CH READ.

Figure 1: Creating a Policy using the Edit/Create a Policy Option—Sample Prompts and User Entries

```
Select DATA ACCESS CONTROL OPTION: ED <Enter> IT/CREATE A POLICY
Select POLICY: LR CH READ
  Are you adding 'LR CH READ' as a new POLICY? No// Y <Enter> (Yes)
   POLICY TYPE: P <Enter> policy
```

Figure 2: DAC Policy Editor—Sample Policy

```
Data Access Control          Sep 27, 2016@13:50:55          Page:    1 of    1
For: <no linked Application Action>

      Name                                          Type      Result            .
1     LR CH READ                                    policy


.          Enter ?? for more actions                                            .
     Add/Remove Members       Expand/Collapse          Test Policy
     Edit a Policy            Inquiry                  Disable a Policy
     Delete a Policy          Functions                Select New Policy
                              Link to Appl Action      Quit
Select Action: Quit//
```

2.2 Add/Remove Members—Adding Rules

The content of the new policy consists of a rule for each possible status of a result. Use the **Add/Remove Members** action to create a rule for the Preliminary (**P**) status. If the rule does *not* exist, you are first allowed to create it and complete the most commonly used fields.

NOTE: The type of Rule is assumed, for policy members.

Finally, a Sequence number *must* be assigned for the new rule within the policy's Members sub-file, as shown in Figure 3.

Figure 3: Creating a Rule for a Policy using the Add/Remove Members Action—Sample Prompts and User Entries (1 of 2)

```
Select Action: Quit// AD <Enter> Add/Remove Members
Select MEMBER: LR CH READ PRELIM
  Are you adding 'LR CH READ PRELIM' as a new POLICY (the 2ND)? No// Y <Enter> (Yes)
NAME: LR CH READ PRELIM// <Enter>
Select TARGET: 1
  Are you adding '1' as a new TARGETS (the 1ST for this POLICY)? No// Y <Enter> (Yes)
  ATTRIBUTE: resultStatus
  VALUE: P
Select TARGET: <Enter>
Select CONDITION: 1
  Are you adding '1' as a new CONDITIONS (the 1ST for this POLICY)? No// Y <Enter> (Yes)
  FUNCTION: DI HAS KEY <Enter>   CONDITION
  VALUE: LRLAB
Select CONDITION: <Enter>
RESULT: P <Enter> PERMIT
DENY MESSAGE: You are not authorized to view preliminary results.
DENY FUNCTION: <Enter>
PERMIT MESSAGE: <Enter>
PERMIT FUNCTION: <Enter>
   MEMBERS SEQUENCE: 1// <Enter>
```

Targets should be defined with a name and value that describe the desired rule. Use the VA FileMan **DI HAS KEY** function to check if the user holds the key named in the VALUE field.

If a rule that already exists is added to a policy, only the **Members Sequence** is prompted.

Create one more similar rule, to handle a result status of Final (**F**), as shown in Figure 4.

Figure 4: Creating a Rule for a Policy using the Add/Remove Members Action—Sample Prompts and User Entries (2 of 2)

```
Select MEMBER: LR CH READ FINAL
  Are you adding 'LR CH READ FINAL' as a new POLICY (the 3RD)? No// Y <Enter> (Yes)
NAME: LR CH READ FINAL// <Enter>
Select TARGET: 1
  Are you adding '1' as a new TARGETS (the 1ST for this POLICY)? No// Y <Enter> (Yes)
  ATTRIBUTE: resultStatus
  VALUE: F
Select TARGET: <Enter>
Select CONDITION: 1
  Are you adding '1' as a new CONDITIONS (the 1ST for this POLICY)? No// Y <Enter> (Yes)
  FUNCTION: DI HAS KEY
  VALUE: PROVIDER
Select CONDITION: 2
  Are you adding '2' as a new CONDITIONS (the 2ND for this POLICY)? No// Y <Enter> (Yes)
  FUNCTION: DI HAS KEY
  VALUE: LRLAB
Select CONDITION: <Enter>
CONDITION CONJUNCTION: ! <Enter> OR
RESULT: P <Enter> PERMIT
DENY MESSAGE: You are not authorized to view lab results.
DENY FUNCTION: <Enter>
PERMIT MESSAGE: <Enter>
PERMIT FUNCTION: <Enter>
   MEMBERS SEQUENCE: 2// <Enter>
```

This rule can be satisfied if the user holds either one of the keys, so two conditions are created; a condition conjunction is then needed to indicate that only one condition needs to be true.

Press **Enter** when finished to exit editing, and redisplay the list showing the hierarchy, as shown in Figure 5.

Figure 5: DAC Policy Editor —Sample Policy with Rules

```
Data Access Control           Sep 27, 2016@13:59:53          Page:    1 of    1
For: <no linked Application Action>

      Name                                              Type      Result          .
1     -LR CH READ                                       policy
2        LR CH READ PRELIM                              rule      PERMIT
3        LR CH READ FINAL                               rule      PERMIT


.          Enter ?? for more actions                                             .
     Add/Remove Members       Expand/Collapse          Test Policy
     Edit a Policy            Inquiry                  Disable a Policy
     Delete a Policy          Functions                Select New Policy
                              Link to Appl Action      Quit
Select Action: Quit//
```

2.3 Edit a Policy

The **Edit a Policy** action invokes a ScreenMan form, which allows reviewing and editing of any rule or policy in the list. All fields are presented for editing, but *not* all need to be completed for every rule or policy. For example, an Attribute Function is usually only defined for the policy, and *not* needed for every rule as well.

Besides its Name, only the Result or Result Function is required for rules or policies, respectively. The Conjunction fields are optional unless more than one target or condition is entered.

Figure 6: Editing a Policy using the Edit a Policy Action—Sample Prompts and User Entries

```
Select Action: Quit// ED <Enter> Edit a Policy
Select Item(s):  (1-3): 3
Loading form to edit #3 ...
```

Figure 7: DAC Policy Editor ScreenMan Form—Review/Edit a Sample Policy (1 of 3)

```
                                  Edit Rule
NAME: LR CH READ FINAL                                              Page 1 of 3
------------------------------------------------------------------------------
              NAME: LR CH READ FINAL
       DESCRIPTION:

  #  Attribute Name                 Value
  1  resultStatus                   F


ATTRIBUTE FUNCTION:
       CONJUNCTION:
______________________________________________________________________________
Exit     Save     Next Page     Refresh

Enter a command or '^' followed by a caption to jump to a specific field.


COMMAND: N                                    Press <PF1>H for help    Insert
```

Figure 8: DAC Policy Editor ScreenMan Form—Review/Edit a Sample Policy (2 of 3)

```
                                  Edit Rule
NAME: LR CH READ FINAL                                              Page 2 of 3
------------------------------------------------------------------------------
  #  Condition                      Parameter
  1  DI HAS KEY                     PROVIDER
  2  DI HAS KEY                     LRLAB


     RESULT: PERMIT
CONJUNCTION: OR
_____________________________________________________________________________
Exit     Save     Next Page     Refresh

Enter a command or '^' followed by a caption to jump to a specific field.


COMMAND: N                                      Press <PF1>H for help    Insert
```

Figure 9: DAC Policy Editor ScreenMan Form—Review/Edit a Sample Policy (3 of 3)

```
                                  Edit Rule
NAME: LR CH READ FINAL                                              Page 3 of 3
------------------------------------------------------------------------------

  ON DENY - FUNCTION:
             MESSAGE: |DIUSRNM| is not authorized to view lab results.


ON PERMIT - FUNCTION:
             MESSAGE:
    AVAILABLE FIELDS:
  Select ADDL FIELDS:



_____________________________________________________________________________
Exit     Save     Next Page     Refresh

Enter a command or '^' followed by a caption to jump to a specific field.


COMMAND: E                                      Press <PF1>H for help    Insert
```

2.3.1 Messages

Advice messages can contain dynamic text, if needed, by using the VA FileMan text bar syntax; in the above example, the name of the user is inserted into the Deny Message when returned. The name of a local variable, or the name of an attribute (i.e., a subscript) in the DIVAL array, can be placed between two vertical bars; when the message is returned, the value of the variable or attribute is inserted into the text.

REF: For the full list of local variables that can be referenced, see "Appendix A—Key DI* Variables."

2.3.2 Available Fields

Not every action on a file requires accessing every field; a developer may want to specify the fields that are appropriate to view or modify. Available Fields can hold a string that can be used as the DR variable in any standard VA FileMan API call. This string is returned to the calling client or application when a Permit result is

obtained. Continuation strings, or specific fields within a sub-file, can be included by creating an entry in the Additional Fields list.

Field strings can be saved at any level of a policy hierarchy, including the associated Application Action. The lowest level in the hierarchy that determined the Permit result takes precedence, and its DR string is returned. It is *recommended* that a default field string be saved at the highest level appropriate, for example with the Action or primary Policy. If exceptions are needed, they should be saved with the rule that grants access under those conditions to override the default string.

2.4 Delete a Policy

A rule or policy can be removed from the current policy tree by using either the **Add/Remove Members** or **Edit a Policy** action. The **Delete a Policy** action completely removes a rule or policy from the POLICY (#1.6) file, and cleans up all pointers to that item.

Figure 10: Deleting a Policy using the Delete a Policy Action—Sample Prompts and User Entries

```
Select Action: Quit// DE <Enter> Delete
Select Item (1-3): 2 <Enter> LR CH READ PRELIM
LR CH READ PRELIM will also be removed as a member from:
   LR CH READ

Are you sure? NO// ?

Enter NO if you only want to remove this item as a member of a policy.

Enter YES to delete this item from the file; it will also be removed as
a Member from its parent policies.

Are you sure? NO// <Enter>
Nothing deleted!
```

2.5 Disable a Policy

The **Disable a Policy** action changes the value of the DISABLE (#.03) field in the POLICY (#1.6) file. If it is set to **YES**, the policy and all of its descendant members are *not* processed when testing or running the policy evaluation API. A disabled policy can also be re-enabled via this action.

Figure 11: Disabling a Policy using the Disable a Policy Action—Sample Prompts and User Entries

```
Select Action: Quit// DI <Enter> Disable Policy
Select Item (1-3): 2 <Enter> LR CH READ PRELIM

WARNING: Disabling a policy will prevent it and ALL its members from being
processed when data access is being evaluated!

DISABLE: Y <Enter> YES
```

A disabled policy is indicated in the editor by parentheses around the name, as shown in Figure 12.

Figure 12: DAC Policy Editor—Sample Disabled Policy

```
Data Access Control           Sep 27, 2016@14:02:55          Page:    1 of    1
For: <no linked Application Action>

      Name                                              Type      Result          .
1     -LR CH READ                                       policy
2        (LR CH READ PRELIM)                            rule      PERMIT
3        LR CH READ FINAL                               rule      PERMIT


.          Enter ?? for more actions                                             .
     Add/Remove Members        Expand/Collapse           Test Policy
     Edit a Policy             Inquiry                   Disable a Policy
     Delete a Policy           Functions                 Select New Policy
                               Link to Appl Action       Quit
Select Action: Quit//
```

2.6 Expand/Collapse

The **Expand/Collapse** action toggles the display to expanding a policy to see all of its rules or collapsing the members. A "+" preceding the policy name indicates that it has members that are collapsed together, as shown in Figure 14.

Figure 13: Collapsing Members of a Policy using the Expand/Collapse Action—Sample Prompts and User Entries

```
Select Action: Quit// EX <Enter> Expand/Collapse
Select Item(s):  (1-3): 1

Re-building the list...
```

Figure 14: DAC Policy Editor—Sample Display of a Policy with Members Collapsed together

```
Data Access Control           Sep 27, 2016@14:05:13          Page:    1 of    1
For: <no linked Application Action>

      Name                                              Type      Result          .
1     +LR CH READ                                       policy


.          Enter ?? for more actions                                             .
     Add/Remove Members        Expand/Collapse           Test Policy
     Edit a Policy             Inquiry                   Disable a Policy
     Delete a Policy           Functions                 Select New Policy
                               Link to Appl Action       Quit
Select Action: Quit//
```

2.7 Inquiry

The **Inquiry** action displays a snapshot of the selected item in VA FileMan captioned format. In addition, the Inquiry action also lists all policies or sets to which the selected item is itself a member.

Figure 15: Using the Inquiry Action—Sample Prompts and User Entries

```
Select Action: Quit// IN <Enter> Inquiry
Select Item (1-3): 3 <Enter> LR CH READ FINAL

NAME: LR CH READ FINAL                  TYPE: rule
  CONDITION CONJUNCTION: OR             RESULT: PERMIT
TARGET: 1                               ATTRIBUTE: resultStatus
  VALUE: F
CONDITION: 1                            FUNCTION: DI HAS KEY
  VALUE: PROVIDER
CONDITION: 2                            FUNCTION: DI HAS KEY
  VALUE: LRLAB
  DENY MESSAGE: |DIUSRNM| is not authorized to view lab results.

MEMBER OF:
   LR CH READ
Enter RETURN to continue or '^' to exit:
```

2.8 Functions

Use the **Functions** action in the Policy Editor to review existing functions, or create new ones. It is *highly recommended* when creating your own functions to document all input and output variables, including source fields, in the Description. VA FileMan reserves the internal entry numbers less than 1000 for its own use, and those functions are *not* editable. You can also assign the function to a policy from within this option, without having to go back into the Edit screens.

Figure 16: Using the Functions Action—Sample Prompts and User Entries

```
Select Action: Quit// F <Enter> Functions
Select POLICY FUNCTION: LRCH ATTRIBUTES
  Are you adding 'LRCH ATTRIBUTES' as a new POLICY FUNCTION (the 8TH)? No// Y <Enter>
(Yes)
   POLICY FUNCTION NUMBER: 1007// <Enter>
   POLICY FUNCTION TYPE: A <Enter> ATTRIBUTE
NAME: LRCH ATTRIBUTES// <Enter>
DISPLAY NAME: Get Lab CH Attributes
EXECUTE CODE: D GETCH^LRZTEST
DESCRIPTION:
  1> <Enter>

Do you want to assign this ATTRIBUTE function to a policy? Y <Enter> YES

Select POLICY: LR CH READ
ATTRIBUTE FUNCTION: LRCH ATTRIBUTES <Enter> ATTRIBUTE

Select POLICY: <Enter>


Select POLICY FUNCTION:
```

2.9 Application Actions

Use the **Link To Appl Action** action in the Policy Editor to assign the policy to an Application Action. A new Application Action can be created here, but it may be helpful to first use the **Set Up Application Actions** option [DIAC ACTIONS] on the Data Access Control menu [DIACCESS]; this option provides full access to the APPLICATION ACTIONS (#1.61) file for creating and managing the actions supported by each application that require data access policies.

REF: For more information on the Data Access Control menu and associated options, see the "Data Access Control Menu [DIACCESS]" section.

Figure 17: Setting Application Actions using the Link to Appl Action—Sample Prompts and User Entries

```
Select Action: Quit// L <Enter> Link to Appl Action

No Application Actions are linked to LR CH READ.

Select APPLICATION ACTION: LRCH READ <Enter> 63.04   read
POLICY: LR CH READ

Select APPLICATION ACTION:
```

Press **Enter** when finished to exit editing and redisplay the list; note the change in the header area as shown in Figure 18.

Figure 18: DAC Policy Editor—Sample Policy with Linked Application Actions

```
Data Access Control          Sep 27, 2016@14:07:21          Page:    1 of    1
For: #63.04, View Chemistry results

      Name                                          Type       Result            .
1    -LR CH READ                                   policy     First Applicable
2       LR CH READ PRELIM                          rule       PERMIT
3       LR CH READ FINAL                           rule       PERMIT


.         Enter ?? for more actions                                              .
     Add/Remove Members        Expand/Collapse           Test Policy
     Edit a Policy             Inquiry                   Disable a Policy
     Delete a Policy           Functions                 Select New Policy
                               Link to Appl Action       Quit
Select Action: Quit//
```

If the current policy being edited is assigned to an action, the File# and Short Description is shown in the header of the Policy Editor, as a reminder of the purpose of this policy (see Figure 18).

2.10 Test Policy

The Policy Editor includes a **Test Policy** action to test a policy. It has an optional trace feature that shows the details of policy execution, including local variables. The output is displayed in the VA FileMan Browser, as shown in Figure 19.

Figure 19: Test Policy Action—Sample Prompts, User Entries, and Output Displayed in the VA FileMan Browser

```
Select Action: Quit// TE <Enter> Test Policy

Enter values to use for testing evaluation of LR CH READ,
either a valid IENS string and/or target attributes.

IENS: <Enter>
ATTRIBUTE: ?

Enter an attribute/value pair for testing evaluation of this policy.
Target attributes used within this policy are:
     resultStatus

ATTRIBUTE: resultStatus
    VALUE: P
ATTRIBUTE: <Enter>

Select Test User: FMUSER,ONE// <Enter>       FM1          TESTER
Show a trace of all policies and rules evaluated? Y <Enter> YES

.                              LR CH READ                                .
DIACT = 5
DIACTN = read
DIENS =
DIFN = 63.04
DIPOL = 1
DIUSR = 1000406
DIVAL("resultStatus") = P

LR CH READ: DIPOL=1 (DIFN=63.04 & DIACTN=read)
   LR READ FINAL: <not a match>
   LR READ PRELIM: resultStatus=P
      DI HAS KEY(LRLAB): 0
      DIRESULT: D
LR CH READ: First Applicable

Result: DENY

DIMSG: 2
FMUSER,ONE is not authorized to view preliminary results.
Please contact Lab staff.


Col>   1 |<PF1>H=Help <PF1>E=Exit| Line>    20 of 20     Screen>     1 of 1
```

2.10.1 Test Utility Output

If you answered **YES** to showing the execution trace, the first thing shown in the VA FileMan Browser is a list of the local DI variables in use. For example:

Figure 20: Test Policy Action—Sample DI Variables in Use

```
DIACT = 5
DIACTN = read
DIENS =
DIFN = 63.04
DIPOL = 1
DIUSR = 1000406
DIVAL("resultStatus") = P
```

Second, the execution trace itself is displayed, indenting each level while drilling down into the hierarchy. For example:

Figure 21: Test Policy Action—Sample Execution Trace

```
LR CH READ: DIPOL=1 (DIFN=63.04 & DIACTN=read)
   LR READ FINAL: <not a match>
   LR READ PRELIM: resultStatus=P
      DI HAS KEY(LRLAB): 0
      DIRESULT: D
LR CH READ: First Applicable
```

The caption is the name of the policy, rule, or function being evaluated; its result is shown to the right of the colon:

- For the top-level policy, the result is the matching policy's IEN in local variable **DIPOL**. In the Policy Editor, this is simply the main policy being edited. In practice, this would be the policy found by looking up the File# and Action in the APPLICATION ACTION (#1.61) file.
- Each member policy or rule shows the targets that matched to apply it or "<not a match>".
- Any condition function evaluated is displayed with the rule's Value in parentheses and the Boolean result after execution.
- When a result is determined for the rule, it is stored in the local variable **DIRESULT**.
- As each policy is exited, it is re-displayed in the list with its Result Function.

The final result of the policy is always shown. For example:

Figure 22: Test Policy Action—Final Result

```
Result: DENY
```

Possible results are:

- **P—**Permit
- **D—**Deny
- **-1—**Error
- **Null—**Unknown

Any messages generated based on the result are displayed next, under **DIMSG** with the number of lines. For example:

Figure 23: Test Policy Action—Sample Messages Generated Based on the Result

```
DIMSG: 2
FMUSER,ONE is not authorized to view preliminary results.
Please contact Lab staff.
```

Similarly, any error messages that occurred are listed under **DIERR** with the error count.

Finally, the Available Fields strings are displayed as **DIFLDS**, if defined, with the name of the rule, policy, or action it came from in parentheses. Any Additional Fields are listed below in the DR array format:

DIFLDS(level,subfile#,n) = field string

2.11 Select New Policy

The **Select New Policy** action allows the user to change the policy that is displayed in the list for editing. A different policy can be selected from the POLICY (#1.6) file, or a brand new one can be created as shown earlier on entry to the Policy Editor. The screen will re-display with the new policy and any currently existing members, ready for editing.

Figure 24: Select New Policy Action—Sample Prompts and User Entries

```
Select Action: Quit// SE <Enter> Select New Policy
Select POLICY: LR MI READ
  Located in the LR (LAB SERVICE) namespace.
  Are you adding 'LR MI READ' as a new POLICY (the 4TH)? No// Y <Enter> (Yes)
   POLICY TYPE: P <Enter> policy
```

Figure 25: DAC Policy Editor—Sample New Policy

```
Data Access Control           Sep 27, 2016@14:10:25          Page:    1 of    1
For: <no linked Application Action>

      Name                                          Type      Result             .
1     LR MI READ                                    policy


.          Enter ?? for more actions                                             .
     Add/Remove Members        Expand/Collapse           Test Policy
     Edit a Policy             Inquiry                   Disable a Policy
     Delete a Policy           Functions                 Select New Policy
                               Link to Appl Action       Quit
Select Action: Quit//
```

2.12 Policy Sets

If more complex policies are needed, policies can be grouped into sets. Sets are simply groups of policies or other sets, and each *must* have its own Result Function. A set can also:

- Have its own Attribute Function and target attributes.
- Return a message.
- Define the Available Fields.
- Be linked to an Application Action for data access evaluation.

For example, Lab may wish to create a set to handle all reads from the LAB DATA (#63) file. A policy similar to LR CH READ could be created for each lab section that might include a target of “labSection” to determine which policy applies to the read request.

Figure 26: DAC Policy Editor—Sample Policy Sets

```
Data Access Control          Sep 27, 2016@14:31:16          Page:    1 of    1
For: #63, View Lab Data file

      Name                                          Type       Result            .
1    -LR READ                                      set        First Applicable
2      +LR CH READ                                 policy     First Applicable
3       LR MI READ                                 policy     First Applicable
4       LR SP READ                                 policy     First Applicable
5       LR CY READ                                 policy     First Applicable
6       LR EM READ                                 policy     First Applicable


.         Enter ?? for more actions                                            .
     Add/Remove Members        Expand/Collapse           Test Policy
     Edit a Policy             Inquiry                   Disable a Policy
     Delete a Policy           Functions                 Select New Policy
                               Link to Appl Action       Quit
Select Action: Quit//
```

3 Data Access Control Menu [DIACCESS]

The DAC also provides a separate Data Access Control menu [DIACCESS]. This menu contains options that allow creation and management of Data Access Control policies for VistA files. This menu contains the following options (listed in the order of appearance on the menu):

- Set Up Application Actions [DIAC ACTIONS]
- Edit/Create an Action Policy [DIAC EDIT]
- Test a Policy [DIAC TEST]
- Disable a Policy [DIAC DISABLE]
- Delete a Policy [DIAC DELETE]
- Print Actions/Policies [DIAC PRINT]
- Policy Functions [DIAC FUNCTIONS]

3.1 Set Up Application Actions

The **Set Up Application Actions** option [DIAC ACTIONS] provides access to the APPLICATION ACTION (#1.61) file, which allows the user to edit an existing or create a new Application Action, as shown in Figure 27.

Figure 27: Set Up Application Actions Option—Sample Prompts and User Entries

```
Select Data Access Control <TEST ACCOUNT> Option: SET UP <Enter> Application Actions
Select APPLICATION ACTION: ??

   Choose from:
   LR READ         63   read
   LRCH READ        63.04   read
   OR DISCONTINUE        100   disc
   OR FLAG        100.008   flag
   OR HOLD        100   hold
   TIU VIEW        8925   view
   YS MH VIEW        601.71   view

        You may enter a new APPLICATION ACTION, if you wish
   The formal unique name of the action, prefaced with the package namespace
   specified in the PACKAGE file, or the letter Z or A.

Select APPLICATION ACTION: LRCH READ <Enter>       63.04   read

NAME: LRCH READ                          FILE#: 63.04
  API NAME: read                         TYPE: READ
  POLICY: LR CH READ
  SHORT DESCRIPTION: View Chemistry results

NAME: LRCH READ// <Enter>
FILE#: 63.04// <Enter>
API NAME: read// <Enter>
TYPE: READ// <Enter>
SHORT DESCRIPTION: View Chemistry results  Replace <Enter>
AVAILABLE FIELDS: <Enter>

Select APPLICATION ACTION:
```

3.2 Edit/Create an Action Policy

The **Edit/Create an Action Policy** option [DIAC EDIT] provides access to the DAC Policy Editor.

Figure 28: Edit/Create an Action Policy Option—Sample Prompts and User Entries

```
Select Data Access Control <TEST ACCOUNT> Option: EDIT <Enter> /Create an Action Policy
Select POLICY: LR CH READ
     1   LR CH READ        policy
     2   LR CH READ FINAL       rule
     3   LR CH READ PRELIM       rule
CHOOSE 1-3: 1 <Enter> LR CH READ     policy

Data Access Control           Mar 06, 2017@12:28:13          Page:    1 of    1
For: #63.04, View Chemistry results

      Name                                              Type       Result
1    -LR CH READ                                       policy     First Applicable
2       LR CH READ FINAL                               rule       PERMIT
3       LR CH READ PRELIM                              rule       PERMIT


           Enter ?? for more actions
     Add/Remove Members        Expand/Collapse           Test Policy
     Edit a Policy             Inquiry                   Disable a Policy
     Delete a Policy           Functions                 Select New Policy
                               Link to Appl Action       Quit
Select Action: Quit//
```

REF: For more information on the Policy Editor and associated actions, see the "Policy Editor" section.

3.3 Test a Policy

The **Test a Policy** option [DIAC TEST] allows users to test a DAC policy.

REF: For more information on the **Test a Policy** option functionality, see the "Test Policy" section.

3.4 Disable a Policy

The **Disable a Policy** option [DIAC DISABLE] provides a way to quickly disable a DAC policy.

REF: For more information on the **Disable a Policy** option functionality, see the "Disable a Policy" section.

3.5 Delete a Policy

The **Delete a Policy** option [DIAC DELETE] allows users to delete a DAC policy.

REF: For more information on the **Delete a Policy** option functionality, see the "Delete a Policy" section.

3.6 Print Actions/Policies

The **Print Actions/Policies** option [DIAC PRINT] allows users to print a summary list of application actions and their associated policies, or details of one policy.

The summary data in Figure 29 is obtained from the APPLICATION ACTION (#1.61) file.

Figure 29: Print Actions/Policies Option—Sample Prompts and User Entries: Summary

```
Select Data Access Control <TEST ACCOUNT> Option: PRINT <Enter> Actions/Policies
Print (S)ummary by Application Action, or (D)etails of a Policy? S <Enter> UMMARY
LIST ACTIONS by: NAME// <Enter>
Start with NAME: FIRST// OR
Go to NAME: LAST// ORZ
  Within NAME, LIST ACTIONS by: <Enter>
DEVICE: ;;9999999 <Enter> NETWORK

APPLICATION ACTION LIST                              MAR 06, 2017@12:06   PAGE 1
NAME                          FILE#      API NAME    POLICY
------------------------------------------------------------------------------

OR DISCONTINUE                100        disc        OR DISCONTINUE
OR FLAG                       100.008    flag        OR FLAG
OR HOLD                       100        hold        OR HOLD ORDERS
```

The detail data in Figure 30 is obtained from the following files:

- POLICY (#1.6)—Primary policy source
- POLICY FUNCTION (#1.62)—Related policy functions
- APPLICATION ACTION (#1.61)—Related policy actions

Figure 30: Print Actions/Policies Option—Sample Prompts and User Entries: Detailed

```
Select Data Access Control <TEST ACCOUNT> Option: PRINT <Enter> Actions/Policies
Print (S)ummary by Application Action, or (D)etails of a Policy? D <Enter> DETAILED
Select POLICY: LR CH READ
     1   LR CH READ        policy
     2   LR CH READ FINAL       rule
     3   LR CH READ PRELIM       rule
CHOOSE 1-3: 1 <Enter> LR CH READ     policy

DEVICE: HOME// ;;999999 <Enter> NETWORK


LR CH READ                                          MAR 06, 2017  12:17    PAGE 1
-------------------------------------------------------------------------------

APPLICATION ACTION: LRCH READ                        TYPE: READ
             FILE#: 63.04                       API NAME: read
 SHORT DESCRIPTION: View Chemistry results

POLICY: LR CH READ                                RESULT: DI FIRST APPLICABLE
ATTRIBUTES: LRCH ATTRIBUTES
TARGETS:
1: labSection = CH
DENY MESSAGE: Please contact Lab staff.
PERMIT FUNCTION: LR ACCESS

RULES:
1: LR CH READ FINAL                               RESULT: PERMIT
   TARGETS:
   1: resultStatus = F
   CONDITIONS (OR):
   1: DI HAS KEY (PROVIDER)
   2: DI HAS KEY (LRLAB)
   DENY MESSAGE: |DIUSRNM| is not authorized to view lab results.

2: LR CH READ PRELIM                              RESULT: PERMIT
   TARGETS:
   1: resultStatus = P
   CONDITIONS:
   1: DI HAS KEY (LRLAB)
   DENY MESSAGE: |DIUSRNM| is not authorized to view preliminary results.

FUNCTION: LRCH ATTRIBUTES                           TYPE: ATTRIBUTE
  DISPLAY NAME: Get Lab CH Attributes
  EXECUTE CODE: Q
 DESCRIPTION:
 placeholder for testing

FUNCTION: DI HAS KEY                                TYPE: CONDITION
  DISPLAY NAME: User Holds Key
  EXECUTE CODE: S Y=$D(^XUSEC(X,+DIUSR))
 DESCRIPTION:
 Requires X = Security Key #19.1 Name
 Returns  Y = $D(^XUSEC(X,+DIUSER))

FUNCTION: LR ACCESS                                 TYPE: OBLIGATION
  DISPLAY NAME: Log access to Lab Data file
  EXECUTE CODE: Q
 DESCRIPTION:
 placeholder for testing

FUNCTION: DI FIRST APPLICABLE                       TYPE: RESULT
  DISPLAY NAME: First Applicable
```

```
  EXECUTE CODE: I DIRESULT'="" S Y=1
 DESCRIPTION:
 Quit when any result is determined


          Set Up Application Actions
          Edit/Create an Action Policy
          Test a Policy
          Disable a Policy
          Delete a Policy
          Print Actions/Policies
          Policy Functions

Select Data Access Control <TEST ACCOUNT> Option:
```

3.7 Policy Functions

The **Policy Functions** option [DIAC FUNCTIONS] provides access to the POLICY FUNCTION (#1.62) file, which allows the user to edit an existing or create a new Policy Function, as shown in Figure 31.

Figure 31: Policy Functions Option—Sample Prompts and User Entries

```
Select Data Access Control <TEST ACCOUNT> Option: POLICY <Enter> Functions
Select POLICY FUNCTION: LRCH ATTRIBUTES <Enter>       ATTRIBUTE

NUMBER: 1002                              NAME: LRCH ATTRIBUTES
  DISPLAY NAME: Get Lab CH Attributes     TYPE: ATTRIBUTE
  EXECUTE CODE: Q
 DESCRIPTION:
 placeholder for testing

NAME: LRCH ATTRIBUTES// <Enter>
DISPLAY NAME: Get Lab CH Attributes  Replace <Enter>
EXECUTE CODE: Q// <Enter>
DESCRIPTION:
placeholder for testing

  Edit? NO// <Enter>

Do you want to assign this ATTRIBUTE function to a policy? <Enter>


Select POLICY FUNCTION:
```

REF: For a list of exported DI* functions, see "Appendix B—Exported DI* Functions."

4 Application Programming Interface (API)

4.1 $$CANDO^DIAC1(): Policy Evaluation

Once a policy has been created and tested, it is ready to be used by the new VA FileMan Web service or within an application's own code. The $$CANDO^DIAC1 API evaluates a policy to determine if the action being attempted is permitted.

If a matching entry exists in the APPLICATION ACTION (#1.61) file for the requested action and specified file or sub-file, its policy is evaluated to determine the user's authorization to access the file and/or record. Policy rules are evaluated in sequence, and processing continues until the stop criteria for the policy is met.

Format

```
$$CANDO^DIAC1(file,iens,action[,user][,.value][,.fields][,msg_root][,err_root])
```

Input Parameters

file:	(Required) A VistA file number or sub-file number.
iens:	(Required/Optional) Standard IENS string indicating internal entry numbers. It is required if evaluating an action on an existing record.
action:	(Required) The API name of the action to be taken on the record; the **file** and **action** parameters should identify an entry in the APPLICATION ACTION (#1.61) file.
user:	(Optional) Pointer to the NEW PERSON (#200) file; defaults to the current value of DUZ if *not* defined.
value(name):	(Optional) Array of additional attribute values to use when evaluating policies, passed by reference in the form: VALUE("name") = "value" The name-value pairs could match target attributes in the policy for supplementing the results of the Attribute Function, or simply be additional values used by other functions or messages.
fields:	(Optional) Local variable that receives output from the call. If the Available Fields have been defined for the application action and its policy returns a permit result, that field string is be returned. If Available Fields have been defined for the policy or rule that determined the result, that string takes precedence over the action's and be returned instead. Additional Fields can also be returned here, as an array of the same name subscripted by the file or sub-file number. **NOTE:** This variable is killed at the beginning of each call.
msg_root:	(Optional) Closed root into which any relevant advice messages is returned. If this parameter is *not* passed, the array is put into nodes descendant from ^TMP("DIMSG",$J).
err_root:	(Optional) Closed root into which the error messages are returned. If this parameter is *not* passed, the array is put into nodes descendant from ^TMP("DIERR",$J).

Output

This Boolean extrinsic function returns the following:

- **1—**If authorization is permitted (Permit).
- **0—**If authorization is denied (Deny).
- **Null—**If authorization *cannot* be determined (i.e., no policies exist in the POLICY [#1.6] file that apply).
- **-1—**Error.

Advice messages can be returned for either a Permit or Deny result. Available Fields are only returned on a Permit.

4.1.1 Examples

To check the current user's authorization to view a chemistry result using our sample Lab policy, a simple call could be made to the API:

```
>S OK=$$CANDO^DIAC1(63.04,"7019779.8679,12345,","read") W !,OK
1
```

A different user may *not* be permitted to view the result, and the message array can show why:

```
>S OK=$$CANDO^DIAC1(63.04,"7019779.8679,12345,","read",1000406,,,"ZZMSG")
>W OK,! ZW ZZMSG
0
ZZMSG(1)="FMUSER,ONE is not authorized to view preliminary results."
ZZMSG(2)="Please contact Lab staff."
```

An incomplete call to the API returns an error:

```
>S OK=$$CANDO^DIAC1(63.04,"7019779.8679,12345,",,,,,"ZZMSG","ZZERR")
>W OK,! ZW ZZERR
-1
ZZERR(1)="The input parameter that identifies the ACTION is missing or invalid."
```

5 Appendix A—Key DI* Variables

These key DI* variables are either passed into the $$CANDO^DIAC1(): Policy Evaluation API, or are set by VA FileMan. They can be referenced in function code as read-only, except for the DIVAL array, which *must* be set by the Attribute Functions.

Table 5: Key DI* Variables

Variables	Description
DIACT	Pointer to the APPLICATION ACTION (#1.61) file of the action that matches DIFN and DIACTN.
DIACTN	The API Name of the action the user is requesting to take on the records, as defined in an APPLICATION ACTION (#1.61) file entry for the file.
DIENS	The IENS string passed into the API.
DIERROR	The closed root of an array in which to return errors instead of ^TMP("DIERR",$J).
DIFCN()	Array holding the Result Function code for each policy in the form DIFCN(IEN)=code, where IEN is the pointer to the POLICY (#1.6) file of the policy being evaluated. The function's default value for a null result is held in DIFCN(IEN,"NULL")=P or D.
DIFLDS	The DR string holding the list of fields that can be accessed on this action; continuation strings or sub-file fields are returned as: DIFLDS(subfile#,n)=string
DIFN	The file or sub-file number where the record being evaluated resides.
DIPOL	Pointer to the POLICY (#1.6) file of the primary policy being tested or evaluated.
DIRESULT	The current result of the policy: • **P—**Permit • **D—**Deny • **-1—**Error • **Null—**Unknown
DITXT	The closed root of an array in which to return messages instead of ^TMP("DIMSG",$J).
DIUSR	Pointer to the NEW PERSON (#200) file of the user requesting to access the data.
DIUSRNM	NAME (#.01) field in the NEW PERSON (#200) file of DIUSR.
DIVAL()	Array holding record attributes as set by the policy's Attribute Function in the form: DIVAL("name") = "value" For comparison to the policy's Target attributes. It can also contain additional values passed into the API or defined during testing.

6 Appendix B—Exported DI* Functions

Figure 1 lists the entries in the POLICY FUNCTION (#1.62) file that are exported with Patch DI*22.2*8 for national/public/supported use.

Figure 32: Exported DI* Functions

```
NUMBER: 1                                NAME: DI FIRST APPLICABLE
  DISPLAY NAME: First Applicable         TYPE: RESULT
  EXECUTE CODE: I DIRESULT'="" S Y=1
 DESCRIPTION:
 Quit when any result is determined

NUMBER: 2                                NAME: DI DENY OVERRIDE
  DISPLAY NAME: Deny Override            TYPE: RESULT
  EXECUTE CODE: I DIRESULT="D" S Y=1
 DESCRIPTION:
 Quit when a Deny result is found

NUMBER: 3                                NAME: DI DENY UNLESS PERMIT
  DISPLAY NAME: Deny Unless Permit       TYPE: RESULT
  EXECUTE CODE: I DIRESULT="P" S Y=1     NULL VALUE: DENY
 DESCRIPTION:
 Quit when a Permit result is found; if no result can be determined,
 return a Deny result

NUMBER: 4                                NAME: DI PERMIT OVERRIDE
  DISPLAY NAME: Permit Override          TYPE: RESULT
  EXECUTE CODE: I DIRESULT="P" S Y=1
 DESCRIPTION:
 Quit when a Permit result is found

NUMBER: 5                                NAME: DI PERMIT UNLESS DENY
  DISPLAY NAME: Permit Unless Deny       TYPE: RESULT
  EXECUTE CODE: I DIRESULT="D" S Y=1     NULL VALUE: PERMIT
 DESCRIPTION:
 Quit when a Deny result is found; if no result can be determined,
 return a Permit result

NUMBER: 6                                NAME: DI HAS KEY
  DISPLAY NAME: User Holds Key           TYPE: CONDITION
  EXECUTE CODE: S Y=$D(^XUSEC(X,DIUSR))
 DESCRIPTION:
 Requires X = Security Key #19.1 Name
 Returns  Y = $D(^XUSEC(X,DIUSR))

NUMBER: 7                                NAME: DI PERSON CLASS
  DISPLAY NAME: User is Class Member     TYPE: CONDITION
  EXECUTE CODE: S Y=$$PCLS^DIACX(X,DIUSR)
 DESCRIPTION:
 Requires X = Person Class #8932.1 IEN or VA Code
 Returns  Y = 1 or 0, if X matches DIUSR's current class assignment
```