**Ambulatory Care Reporting Project (ACRP)**

**Interface Toolkit (AIT)**

**January 1998**

**Introduction**

**Application Programmer Interfaces**

56 - SDOE Get Diagnoses

58 - SDOE Get Providers

61 - SDOE Get Procedures

63 - SDOE Assigned a Provider

64 - SDOE Assigned a Diagnosis

65 - SDOE Assigned a Procedure

69 - SDOE Find Provider

70 - SDOE Find Diagnosis

71 - SDOE Find Procedure

72 - SDOE Find First Standalone

73 - SDOE Get Primary Diagnosis

74 - SDOE Find First Encounter

75 - SDOE Find Last Standalone

76 - SDOE Get General Data

78 - SDOE Parse General Data

79 - SDQ Open

80 - SDQ Close

81 - SDQ Patient

82 - SDQ Date Range

83 - SDQ Filter

84 - SDQ Visit

85 - SDQ Index Name

86 - SDQ EOF

87 - SDQ BOF

88 - SDQ Active Status

89 - SDQ Count

90 - SDQ First

91 - SDQ Last

92 - SDQ Next

93 - SDQ Prior

94 - SDQ Refresh

95 - SDQ Get Current Entry ID

98 - SDOE Get Zero Node

99 - SDQ Scan

100 - SDQ Scan Callback

101 - SDQ Error Check

**Remote Procedure Calls**

SDOE Assigned a Diagnosis

SDOE Assigned a Procedure

SDOE Assigned a Provider

SDOE Find Diagnosis

SDOE Find First Encounter

SDOE Find First Standalone

SDOE Find Last Standalone

SDOE Find Procedure

SDOE Find Provider

SDOE Get Diagnoses

SDOE Get General Data

SDOE Get Primary Diagnosis

SDOE Get Procedures

SDOE Get Providers

SDOE Get Zero Node

SDOE List Encounters for Dates

SDOE List Encounters for PAT

SDOE List Encounters for Visit

SDOE Parse General Data

**Error Processing**

Structure

Error Arrays

AIT Error Processing Tools

Checking for Errors

Debugging

### Appendix - Argument Definitions

Action

Begin Date/Time

CPT IEN

Diagnosis IEN

Direction

Encounter Data

Encounter IEN

Encounter Parse Format

Encounter Parsed Data

Encounter Query Active Status

Encounter Query Filter

Encounter Query Handle

Encounter Query Index

End Date/Time

Error Array

List of V CPT Entries

List of V POV Entries

List of V PROVIDER Entries

Patient ID

Practitioner ID

Scan Callback Logic

Search Flags

Visit IEN

Introduction

The ACRP Interface Toolkit (AIT) is a set of programmer tools that provides access to outpatient encounter data. This initial version contains Application Programmer Interfaces (APIs) and Remote Procedure Calls (RPCs) that provide access to procedure, diagnosis, provider, and general data related to an encounter. It is hoped that in a future version of the AIT, Delphi objects, components, and DLLs will be provided as well.

This AIT provides Class I packages, Class III software, and other local code with one highly structured interface to the encounter data.

Application Programmer Interfaces

**ID:** 56

**Name:** SDOE Get Diagnoses

**Declaration:** GETDX^SDOE( encounter, dx\_list [,errors] )

**Description:** This procedure returns a subscripted array of diagnoses for an

encounter.

The array subscripts are arbitrary integer numbers.

Each array entry equals the zeroth node of a record in the V POV file.

The top level, non-subscripted value of the array variable is the count of the number of V POV entries in the array.

NOTE 1: For encounters before 10/1/96, only scheduling data in the

OUTPATIENT DIAGNOSIS (#409.43) file may exist. It will only exist if the site required diagnoses as part of the check-out process.

This API will attempt to find this “old” data, reformat the data to meet the V POV structure, and return the list of diagnoses as stated in the above description. (Only the diagnosis code internal entry number is available for “old” encounters).

NOTE 2: Currently, each array entry corresponds to an entry in the V

POV file. In the future, it is possible that **V***IST***A** will not use V POV as the source of diagnostic data. However, if another source is used, this API will reformat that source and present the data in the V POV format so as not to affect the use of the API by calling applications.

**Arguments:** encounter Encounter IEN

dx\_list List of V POV Entries

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** SDOE GET DIAGNOSES

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** N DXLIST,I

>>> D GETDX^SDOE(4592,"DXLIST")

S I=0

F S I=$O(DXLIST(I)) Q:'I W !,I,":",DXLIST(I)

W !,"Count: ",+$G(DXLIST)

--> 370: 97^101^459^192^^^^^^^^P

Count: 1

**ID:** 58

**Name:** SDOE Get Providers

**Declaration:** GETPRV^SDOE( encounter, provider\_list [,errors] )

**Description:** This procedure returns a subscripted array of providers for an

encounter.

The array subscripts are arbitrary integer numbers.

Each array entry equals the zeroth node of a record in the V PROVIDER file.

The top level, non-subscripted value of the array variable is the count of the number of V PROVIDER entries in the array.

NOTE 1: For encounters before 10/1/96, only scheduling data in the

OUTPATIENT PROVIDER (#409.44) file may exist. It will only exist if the site required provider as part of the check-out process.

This API will attempt to find this “old” data, reformat the data to meet the V PROVIDER structure, and return the list of providers as stated in the above description. (Only the provider internal entry number is available for “old” encounters).

NOTE 2: Currently, each array entry corresponds to an entry in the V

PROVIDER file. In the future, it is possible that **V***IST***A** will not use V PROVIDER as the source of provider data. However, if another source is used, this API will reformat that source and present the data in the V PROVIDER format so as not to affect the use of the API by calling applications.

**Arguments:** encounter Encounter IEN

provider\_list List of V PROVIDER Entries

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** SDOE GET PROVIDERS

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** N PRVLIST,I

>>> D GETPRV^SDOE(4592,"PRVLIST")

S I=0

F S I=$O(PRVLIST(IEN)) Q:'I W !,I,":",PRVLIST(I)

W !,"Count: ",+$G(PRVLIST)

--> 284: 11344^706^407^P^^11

Count: 1

**ID:** 61

**Name:** SDOE Get ProcedureS

**Declaration:** GETCPT^SDOE( encounter, cpt\_list [,errors] )

**Description:** This procedure returns a subscripted array of CPT information for an encounter.

The subscripts are arbitrary integers; however the first subscript is currently the V CPT IEN. The subscript should be treated as an arbitrary integer to allow for possible future changes in the data source. The subscripts and data contained in the array correspond to the V CPT file's DD structure.

The top level, non-subscripted value of the array variable is the count of the number of V CPT entries associated with the encounter.

NOTE 1: For encounters before 10/1/96, only scheduling data in the

SCHEDULING VISITS (#409.5) file may exist. It will only

exist if the site required procedures as part of the check-out process.

This API will attempt to find this “old” data, reformat the data to meet the V CPT structure, and return the list of procedures as stated in the above description. (Only the CPT code internal entry number and count are available for “old” encounters).

NOTE 2: Currently, each array entry corresponds to an entry in the V

CPT file. In the future, it is possible that **V***IST***A** will not use V CPT as the source of procedure data. However, if another source is used, this API will reformat that source and present the data in the V CPT format so as not to affect the use of the API by calling applications.

**Arguments:** encounter Encounter IEN

cpt\_list List of V CPT Entries

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** SDOE GET PROCEDURES

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** N CPTLIST,I,J

>>> D GETCPT^SDOE(8676,"CPTLIST")

ZW CPTLIST

--> CPTLIST=2

CPTLIST(352)=10060^706^407^69^^^^^^^^^^^^1

CPTLIST(352,0)=10060^706^407^69^^^^^^^^^^^^1

CPTLIST(352,12)=^^^1934

CPTLIST(352,801)=^13-A 1312;

CPTLIST(352,812)=^16^11

CPTLIST(2043)=61510^717^488^91^^^^^^^^^^^^1

CPTLIST(2043,0)=61510^717^488^91^^^^^^^^^^^^1

CPTLIST(2043,1,0)=^9000010.181P^1^1

CPTLIST(2043,1,1,0)=16

CPTLIST(2043,1,"B",16,1)=

CPTLIST(2043,12)=^^^9

CPTLIST(2043,801)=^9-A 11723;

CPTLIST(2043,812)=^413^9

**ID:** 63

**Name:** SDOE Assigned a Provider

**Declaration:** $$PRV^SDOE( encounter [,errors] )

**Description:** This function returns a boolean indicator on whether at least one

provider has been associated with an encounter.

**Arguments:** encounter Encounter IEN

errors Error Array [optional]

**Return Values:** 1 - Yes, at least one provider is associated with encounter

0 - No, no providers are associated with encounter

**Related RPC:** SDOE ASSIGNED A PROVIDER

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** >>> W $$PRV^SDOE(4592)

--> 1**ID:** 64

**Name:** SDOE Assigned a Diagnosis

**Declaration:** $$DX^SDOE( encounter [,errors] )

**Description:** This function returns a boolean indicator on whether at least one

diagnoses has been associated with an encounter.

**Arguments:** encounter Encounter IEN

errors Error Array [optional]

**Return Values:** 1 - Yes, at least one diagnosis is associated with encounter

0 - No, no diagnoses are associated with encounter

**Related RPC:** SDOE ASSIGNED A DIAGNOSIS

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** >>> W $$DX^SDOE(4592)

--> 1**ID:** 65

**Name:** SDOE Assigned a Procedure

**Declaration:** $$CPT^SDOE( encounter [,errors] )

**Description:** This function returns a boolean indicator on whether at least one

procedure has been associated with an encounter.

**Arguments:** encounter Encounter IEN

errors Error Array [optional]

**Return Values:** 1 - Yes, at least one procedure is associated with encounter

0 - No, no procedures are associated with encounter

**Related RPC:** SDOE ASSIGNED PROCEDURE

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** >>> W $$CPT^SDOE(4592)

--> 1**ID:** 69

**Name:** SDOE Find Provider

**Declaration:** $$FINDPRV^SDOE( encounter, provider [,errors] )

**Description:** This function returns a boolean indicator on whether a specific provider

is associated with an encounter.

**Arguments:** encounter Encounter IEN

provider Practitioner ID

errors Error Array [optional]

**Return Values:** 1 - Yes, specific provider is associated with encounter

0 - No, provider is not associated with encounter

**Related RPC:** SDOE FIND PROVIDER

**Error Codes:** 4,096,800.001 Invalid Encounter ID

4,096,800.003 Invalid Provider ID

**Example:** >>> W $$FINDPRV^SDOE(4592,990)

--> 1**ID:** 70

**Name:** SDOE Find Diagnosis

**Declaration:** $$FINDDX^SDOE( encounter, diagnosis [,errors] )

**Description:** This function returns a boolean indicator on whether a specific

diagnosis is associated with an encounter.

**Arguments:** encounter Encounter IEN

diagnosis Diagnosis IEN

errors Error Array [optional]

**Return Values:** 1 - Yes, specific diagnosis is associated with encounter

0 - No, diagnosis is not associated with encounter

**Related RPC:** SDOE FIND DIAGNOSIS

**Error Codes:** 4,096,800.001 Invalid Encounter ID

4,096,800.004 Invalid Diagnosis ID

**Example:** >>> W $$FINDDX^SDOE(4592,35)

--> 1**ID:** 71

**Name:** SDOE FIND PROCEDURE

**Declaration:** $$FINDCPT^SDOE( encounter, cpt [,errors] )

**Description:** This function returns a boolean indicator on whether a specific

procedure is associated with an encounter.

**Arguments:** encounter Encounter IEN

cpt CPT IEN

errors Error Array [optional]

**Return Values:** 1 - Yes, specific procedure is associated with encounter

0 - No, procedure is not associated with encounter

**Related RPC:** SDOE FIND PROCEDURE

**Error Codes:** 4,096,800.001 Invalid Encounter ID

4,096,800.005 Invalid CPT ID

**Example:** >>> W $$FINDCPT^SDOE(4592,10061)

--> 1**ID:** 72

**Name:** SDOE Find First Standalone

**Declaration:** $$EXAE^SDOE( dfn, begin\_date, end\_date [,flags][,errors] )

**Description:** This function returns the internal entry number of an Outpatient

Encounter file (#409.68) entry for the first standalone add/edit for a

patient in a specified date range.

Use same date for begin and end dates for specific (single) date check.

Standalone encounter is an encounter with no parent and the

originating process is “Stop Code Addition”.

**Arguments:** dfn Patient ID

begin\_date Begin Date/Time

end\_date End Date/Time

flags Search Flags

errors Error Array [optional]

**Return Values:** <pointer> Outpatient Encounter IEN for first standalone

encounter found in date range

<null> if no encounter exists

**Related RPC:** SDOE FIND FIRST STANDALONE

**Error Codes:** 4,096,800.002 Invalid Patient ID

4,096,800.022 Invalid Date Range

**Example:** >>> W $$EXAE^SDOE(101,2970501,2970601,“C”)

--> 4501**ID:** 73

**Name:** SDOE Get Primary Diagnosis

**Declaration:** $$GETPDX^SDOE( encounter [,errors] )

**Description:** This function returns the internal entry number of the primary

diagnosis code (^ICD9) for an encounter.

NOTE: For encounters before 10/1/96, this function will always return

0. This primary diagnosis was not retrieved nor stored by the system for these “old” encounters.

**Arguments:** encounter Encounter IEN

errors Error Array [optional]

**Return Values:** <pointer> ien to ^ICD9 for primary dx

0 no primary dx found for encounter

**Related RPC:** SDOE GET PRIMARY DIAGNOSIS

**Error Codes:** 4,096,800.001 Invalid Encounter ID

4,096,800.025 Duplicate Primary Diagnosis

**Example:** >>> W $$GETPDX^SDOE(4592)

--> 35**ID:** 74

**Name:** SDOE Find First Encounter

**Declaration:** $$EXOE^SDOE( dfn, begin\_date, end\_date [,flags][,errors] )

**Description:** This function returns the internal entry number of an OUTPATIENT

ENCOUNTER file (#409.68) entry for the first encounter for a patient in

a specified date range.

**Arguments:** dfn Patient ID

begin\_date Begin Date/Time

end\_date End Date/Time

flags Search Flags

errors Error Array [optional]

**Return Values:** <pointer> Outpatient Encounter ID for first encounter found in

date range

<null> if no encounter exists

**Related RPC:** SDOE FIND FIRST ENCOUNTER

**Error Codes:** 4,096,800.002 Invalid Patient ID

4,096,800.022 Invalid Date Range

**Example:** >>> W $$EXOE^SDOE(101,2970501,2970601,“C”)

--> 4504**ID:** 75

**Name:** SDOE Find Last Standalone

**Declaration:** $$GETLAST^SDOE( dfn, begin\_date [,flags][,errors] )

**Description:** This function returns the internal entry number of an Outpatient

Encounter file (#409.68) entry for the last standalone add/edit for a

patient. All encounters from the specified begin date to the current

date are used to find this last standalone.

Standalone encounter is an encounter with no parent and the

originating process is “Stop Code Addition”.

**Arguments:** dfn Patient ID

begin\_date Begin Date/Time

flags Search Flags

errors Error Array [optional]

**Return Values:** <pointer> Outpatient Encounter ID for last standalone encounter

found after date

<null> if no encounter exists

**Related RPC:** SDOE FIND LAST STANDALONE

**Error Codes:** 4,096,800.002 Invalid Patient ID

4,096,800.022 Invalid Date Range

**Example:** >>> W $$GETLAST^SDOE(101,2970414,“C”)

--> 4559**ID:** 76

**Name:** SDOE Get General Data

**Declaration:** GETGEN^SDOE( encounter, encounter\_data [,errors] )

**Description:** This procedure returns the zeroth and other nodes of an outpatient

encounter entry.

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the

zeroth node are returned.

For detail information regarding the fields, see the data dictionary for the OUTPATIENT ENCOUNTER file (#409.68).

**Arguments:** encounter Encounter IEN

encounter\_data Encounter Data

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** SDOE GET GENERAL DATA

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** N DATA,NODE,XXERR

>>> D GETGEN^SDOE(4592,"DATA","XXERR")

S NODE="" F S NODE=$O(DATA(NODE)) Q:NODE="" D

. W !,"Node: ",NODE," = ",DATA(SDNODE)

--> Node: 0 = 2970602.08^706^144^62^407^^

2970805.1107^^^9^1^2^10**ID:** 78

**Name:** SDOE Parse General Data

**Declaration:** PARSE^SDOE( .encounter\_data, format, parsed\_data [,errors] )

**Description:** This procedure will parse the data returned by the “SDOE GET

GENERAL DATA” API into individual field nodes.

The parser will either use internal or external values for the field node

values. The developer specifies internal/external via a parameter.

**Arguments:** encounter\_data Encounter Data

format Encounter Parse Format

parsed\_data Encounter Parsed Data

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** SDOE PARSE GENERAL DATA

**Error Codes:** 4,096,800.023 Invalid Parse Format

4,096,800.024 No Data to Parse

**Example:** N DATA,NODE,PARSED

D GETGEN^SDOE(4592,"DATA")

>>> D PARSE^SDOE(.DATA,"EXTERNAL","PARSE")

S NODE="" F S NODE=$O(PARSE(NODE)) Q:NODE="" D

. W !,"Node: " NODE," ==> ",PARSE(NODE)

--> Node: .01 ==> Jun 02, 1997@08:00

Node: .02 ==> DAVIS,SUE

Node: .03 ==> DERMATOLOGY

Node: .04 ==> DERMATOLOGY

Node: .05 ==> Jun 02, 1997@08:00

Node: .06 ==>

Node: .07 ==> Aug 05, 1997@11:07

Node: .08 ==> APPOINTMENT

Node: .1 ==> REGULAR

Node: .11 ==> TROY

Node: .12 ==> CHECKED OUT

Node: .13 ==> NSC**ID:** 79

**Name:** SDQ Open

**Declaration:** OPEN^SDQ( .query [,errors] )

**Description:** This method is used by a developer to initialize a Query Object and

obtain a query handle.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.109 Invalid Query Property

**Example:** EN N QUERY

>>> D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 80

**Name:** SDQ Close

**Declaration:** CLOSE^SDQ( .query [,errors] )

**Description:** This method is used by a developer to close a Query Object.

NOTE: The Encounter Query handle is set to null as a result of a

successful call to SDQ CLOSE.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

>>> D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 81

**Name:** SDQ Patient

**Declaration:** PAT^SDQ( query, .dfn, action [,errors] )

**Description:** This method is used to set and retrieve the Patient property of a Query

Object.

In order to activate a query, this property must be set if the Index

Name property is either PATIENT or PATIENT/DATE.

**Arguments:** query Encounter Query Handle

dfn Patient ID

action Action

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.002 Invalid Patient ID

4,096,800.101 Invalid Query Object Handle

4,096,800.106 Active Query

4,096,800.108 Invalid Property Action

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

>>> IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CALLBACK^XXSCAN(Y,

Y0,.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 82

**Name:** SDQ Date Range

**Declaration:** DATE^SDQ( query, .begin\_date, .end\_date, action [,errors] )

**Description:** This method is used to set and retrieve the Date Range property of a

Query Object.

In order to activate a query, this property must be set if the Index

Name property is either DATE/TIME or PATIENT/DATE.

NOTE: During a “set” action, a developer can pass in a

“Begin Date/Time” of zero. This will be converted to January 1, 1990 (VA FileMan internal format of 2900101).

**Arguments:** query Encounter Query Handle

begin\_date Begin Date/Time

end\_date End Date/Time

action Action

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.022 Invalid Date Range

4,096,800.101 Invalid Query Object Handle

4,096,800.106 Active Query

4,096,800.108 Invalid Property Action

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

>>> IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 83

**Name:** SDQ Filter

**Declaration:** FILTER^SDQ( query, .filter, action [,errors] )

**Description:** This method is used to set and retrieve the Filter property of a Query

Object.

The following local variables are available at run-time:

Y current encounter entry number

Y0 zeroth node of current encounter entry (only supported fields)

NOTE: If the developer plans to use the SDQ SCAN method of the

query object, it is more efficient to place any “filter-type” logic in the Callback property logic. It is still valid to set the Filter property and call the SDQ SCAN method. However, the method will execute faster if the Callback property logic contains the filter logic.

**Arguments:** query Encounter Query Handle

filter Encounter Query Filter

action Action

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.104 Invalid Filter

4,096,800.106 Active Query

4,096,800.108 Invalid Property Action

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

>>> IF ‘$$ERRCHK^SDQUT() D FILTER^SDQ(.QUERY,"IF $P(Y0,U,8)=1",

"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 84

**Name:** SDQ Visit

**Declaration:** VISIT^SDQ( query, .visit, action [,errors] )

**Description:** This method is used to set and retrieve the Visit property of a Query

Object.

In order to activate a query, this property must be set if the Index

Name property is set to VISIT.

**Arguments:** query Encounter Query Handle

visit Visit IEN

action Action

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 1,509,000.001 Invalid Visit IEN

4,096,800.101 Invalid Query Object Handle

4,096,800.106 Active Query

4,096,800.108 Invalid Property Action

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"VISIT","SET")

>>> IF ‘$$ERRCHK^SDQUT() D VISIT^SDQ(.QUERY,875833,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 85

**Name:** SDQ Index Name

**Declaration:** INDEX^SDQ( query, .index, action [,errors] )

**Description:** This method is used to set and retrieve the Index Name property of a

Query Object.

Valid Values

PATIENT

PATIENT/DATE

DATE/TIME

VISIT

**Arguments:** query Encounter Query Handle

index Encounter Query Index

action Action

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.105 Invalid Index

4,096,800.106 Active Query

4,096,800.108 Invalid Property Action

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

>>> IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 86

**Name:** SDQ EOF

**Declaration:** $$EOF^SDQ( query [,errors] )

**Description:** This function returns a boolean that indicates whether the Query

Object cursor is positioned on the last encounter record in the result

set.

This is a read-only, run-time-only property.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** 1 Yes, query cursor is at the last record or no records exist for

query

0 No, query cursor is not at last record

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2971001,2971231,"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D FIRST^SDQ(.QUERY)

>>> IF ‘$$ERRCHK^SDQUT() F Q:$$EOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

. D NEXT^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

;

PROCESS(SDQ,SDERR) ; -- do some process

N SDOE

S SDOE=+$$GETENTRY^SDQ(.SDQ,$G(SDERR)) ;

W !,SDOE," >>>",$$GETOE^SDOEOE(.SDOE,$G(SDERR))

Q

**ID:** 87

**Name:** SDQ BOF

**Declaration:** $$BOF^SDQ( query [,errors] )

**Description:** This function returns a boolean that indicates whether the Query

Object cursor is positioned on the first encounter record in the result

set.

This is a read-only, run-time-only property.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** 1 Yes, query cursor is at the first record or no records exist for

query

0 No, query cursor is not at first record

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2961215,2961215.2359,,

"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D LAST^SDQ(.QUERY)

>>> IF ‘$$ERRCHK^SDQUT() F Q:$$BOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

. D PRIOR^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

;

PROCESS(SDQ,SDERR) ; -- do some process

N SDOE

S SDOE=+$$GETENTRY^SDQ(.SDQ,$G(SDERR)) ;

W !,SDOE," >>>",$$GETOE^SDOEOE(.SDOE,$G(SDERR))

Q

**ID:** 88

**Name:** SDQ Active Status

**Declaration:** ACTIVE^SDQ( query, .status, action [,errors] )

**Description:** This method is used to set and retrieve the Active Status property of a

Query Object.

Setting this property from FALSE to TRUE will cause the query to

execute producing a new result set. The query cursor will be positioned

on the first record in the result set.

Setting this property from TRUE to FALSE will cause the result set to

be removed.

Other methods like SDQ REFRESH and SDQ NEXT will also return an

error if this property is set to FALSE.

Finally, setting the status to TRUE causes validity checks to be

invoked. These checks determine whether all the necessary query

properties are set correctly. If they are not set, then the status will not

be changed to TRUE and an Invalid Query Property error is

returned in the error array parameter.

**Arguments:** query Encounter Query Handle

status Encounter Query Active Status

action Action

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.103 Invalid Active Status

4,096,800.108 Invalid Property Action

4,096,800.109 Invalid Query Property

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

>>> IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 89

**Name:** SDQ Count

**Declaration:** $$COUNT^SDQ( query [,errors] )

**Description:** This function returns the number of encounter records in the Query

Object's result set.

This function takes a variable amount of time to execute depending on

the result set produced by the query and whether all entries have been

previously accessed by the query object

This is a read-only, run-time-only property.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** <number> count of records in query's result set

0 no records in the query's result set

null invalid query or query not active

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

>>> IF ‘$$ERRCHK^SDQUT() W !,"Count: ",$$COUNT^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

--> Count: 5

**ID:** 90

**Name:** SDQ First

**Declaration:** FIRST^SDQ( query [,errors] )

**Description:** This method positions the query cursor at the first encounter record in

the Query Object's result set.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2971215,2971221,"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

>>> IF ‘$$ERRCHK^SDQUT() D FIRST^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() F Q:$$EOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

. D NEXT^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

;

PROCESS(SDQ,SDERR) ; -- do some process

N SDOE

S SDOE=+$$GETENTRY^SDQ(.SDQ,$G(SDERR)) ;

W !,SDOE," >>>",$$GETOE^SDOEOE(.SDOE,$G(SDERR))

Q

**ID:** 91

**Name:** SDQ Last

**Declaration:** LAST^SDQ( query [,errors] )

**Description:** This method positions the query cursor at the last encounter record in

the Query Object's result set.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2960415,2970415,"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

>>> IF ‘$$ERRCHK^SDQUT() D LAST^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() F Q:$$BOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

. D PRIOR^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

;

PROCESS(SDQ,SDERR) ; -- do some process

N SDOE

S SDOE=+$$GETENTRY^SDQ(.SDQ,$G(SDERR)) ;

W !,SDOE," >>>",$$GETOE^SDOEOE(.SDOE,$G(SDERR))

Q

**ID:** 92

**Name:** SDQ Next

**Declaration:** NEXT^SDQ( query [,errors] )

**Description:** This method positions the query cursor at the next encounter record in

the Query Object's result set.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

4,096,800.111 End of File

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970615,"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D FIRST^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() F Q:$$EOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

>>> . D NEXT^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY) Q

;

PROCESS(SDQ,SDERR) ; -- do some process

N SDOE

S SDOE=+$$GETENTRY^SDQ(.SDQ,$G(SDERR)) ;

W !,SDOE," >>>",$$GETOE^SDOEOE(.SDOE,$G(SDERR))

Q

**ID:** 93

**Name:** SDQ Prior

**Declaration:** PRIOR^SDQ( query [,errors] )

**Description:** This method positions the query cursor at the prior encounter record in

the Query Object's result set.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

4,096,800.110 Beginning of File

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970101,2971231,"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D LAST^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() F Q:$$BOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

>>> . D PRIOR^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

;

PROCESS(SDQ,SDERR) ; -- do some process

N SDOE

S SDOE=+$$GETENTRY^SDQ(.SDQ,$G(SDERR)) ;

W !,SDOE," >>>",$$GETOE^SDOEOE(.SDOE,$G(SDERR))

Q

**ID:** 94

**Name:** SDQ Refresh

**Declaration:** REFRESH^SDQ( query [,errors] )

**Description:** This method causes the Query Object to be re-executed and produce a

new result set based on the most recent data in the database.

This method does the same thing as setting the Active Status property

from TRUE to FALSE and then FALSE to TRUE.

Also, this method positions the query cursor at the first encounter

record in the result set.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

IF ‘$$ERRCHK^SDQUT() D CHANGE^XXEDIT

>>> IF ‘$$ERRCHK^SDQUT() D REFRESH^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 95

**Name:** SDQ Get Current Entry ID

**Declaration:** $$GETENTRY^SDQ( query [,errors] )

**Description:** This function returns the internal entry number to the OUTPATIENT

ENCOUNTER file (#409.68) for the encounter record at the current

Query Object cursor position.

This is a read-only property.

**Arguments:** query Encounter Query Handle

errors Error Array [optional]

**Return Values:** <pointer> ID for entry

<null> if no entries in result set

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2961201,2961231.2359,

"SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D LAST^SDQ(.QUERY)

>>> W !,"Entry: ",$$GETENTRY^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

Q

--> Entry: 3838221

**ID:** 98

**Name:** SDOE Get Zero Node

**Declaration:** $$GETOE^SDOE( encounter [,errors] )

**Description:** This function returns the zeroth node of an outpatient encounter.

NOTE: Currently (7/97), only fields .01 thru .08 and .1 thru .13 of the

zeroth are returned. Other nodes and fields are not supported.

For detail information regarding the fields, see the data dictionary for the OUTPATIENT ENCOUNTER file (#409.68).

**Arguments:** encounter Encounter IEN

errors Error Array [optional]

**Return Values:** Zeroth node of outpatient encounter or null

NOTE: Only supported fields are returned. Those fields not

supported/returned are null.

**Related RPC:** SDOE GET ZERO NODE

**Error Codes:** 4,096,800.001 Invalid Encounter ID

**Example:** >>> W $$GETOE^SDOE(4592)

--> 2970602.08^706^144^62^407^^2970805.1107^1^^9^1^2^10**ID:** 99

**Name:** SDQ Scan

**Declaration:** SCAN^SDQ( query [,direction] [,errors] )

**Description:** This procedure scans encounter records that meet criteria defined in

the Query Object. For each encounter record meeting the criteria, the

Callback property is executed. (See the API definition for SDQ SCAN

CALLBACK for more information.)

NOTE: This procedure tends to be faster than using the SDQ NEXT/

SDQ EOF scan approach when many encounter records must be processed.

NOTE: Default scan direction is FORWARD.

**Arguments:** query Encounter Query Handle

scan\_direction Direction [optional]

errors Error Array [optional]

**Return Values:** Not applicable for procedures

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.102 Inactive Query

4,096,800.112 No Scan Callback Property

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

>>> IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 100

**Name:** SDQ Scan Callback

**Declaration:** SCANCB^SDQ( query, .callback, action [,errors] )

**Description:** This method is used to set and retrieve the Callback property of a

Query Object.

This property is application M code executed by the query object SDQ

SCAN method for each entry found in the encounter record scan

process.

The following local variables are available at run-time:

Y current encounter entry number

Y0 zeroth node of current encounter entry (only supported fields)

**Arguments:** query Encounter Query Handle

callback Scan Callback Logic

action Action

errors Error Array [optional]

**Return Values:** Not applicable for properties

**Related RPC:** Not applicable

**Error Codes:** 4,096,800.101 Invalid Query Object Handle

4,096,800.106 Active Query

4,096,800.108 Invalid Property Action

4,096,800.113 Invalid Scan Callback

**Example:** EN N QUERY

D OPEN^SDQ(.QUERY)

IF ‘$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF ‘$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF ‘$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970501,2970601,"SET")

>>> IF ‘$$ERRCHK^SDQUT() D SCANCB^SDQ(.QUERY,"D CB^XXSCAN(Y,Y0,

.SDSTOP)","SET")

IF ‘$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF ‘$$ERRCHK^SDQUT() D SCAN^SDQ(.QUERY,"FORWARD")

D CLOSE^SDQ(.QUERY)

Q

;

CB(SDOE,SDOE0,SDSTOP) ; -- callback logic

W !,SDOE," >>>",SDOE0

Q

**ID:** 101

**Name:** SDQ error check

**Declaration:** $$ERRCHK^SDQUT( [errors] )

**Description:** This general utility function returns a boolean that indicates whether

the current error array contains any errors.

The current error array is the array indicated by the optional “error”

parameter. If this “error” parameter is not specified, the standard

^TMP(“DIERR”,$J) is used.

**Arguments:** errors Error Array [optional]

**Return Values:** 1 Yes, at least one error is in the error array

0 No, no errors are in the error array

**Related RPC:** Not applicable

**Error Codes:** None

**Example:** >>> W $$ERRCHK^SDQUT()

--> 1

Remote Procedure Calls

**Name:** SDOE ASSIGNED A DIAGONSIS

**Description:** This Remote Procedure Call (RPC) returns a boolean indicator on

whether at least one diagnoses has been associated with an encounter.

**Related API:** 64 - SDOE ASSIGNED A DIAGNOSIS

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE ASSIGNED A DIAGNOSIS';

broker.Param[0].Value := txtEncounter.Text;

broker.Param[0].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE ASSIGNED A PROCEDURE

**Description:** This Remote Procedure Call (RPC) returns a boolean indicator on

whether at least one procedure has been associated with an encounter.

**Related API:** 65 - SDOE ASSIGNED A PROCEDURE

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE ASSIGNED A PROCEDURE'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal; broker.Call; memResults.Lines := broker.Results;

end;

**Name:** SDOE ASSIGNED A PROVIDER

**Description:** This Remote Procedure Call (RPC) returns a boolean indicator on

whether at least one provider has been associated with an encounter.

**Related API:** 63 - SDOE ASSIGNED A PROVIDER

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE ASSIGNED A PROVIDER';

broker.Param[0].Value := txtEncounter.Text;

broker.Param[0].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE FIND DIAGNOSIS

**Description:** This Remote Procedure Call (RPC) returns a boolean indicator on

whether a specific diagnosis is associated with an encounter.

**Related API:** 70 - SDOE FIND DIAGNOSIS

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

1 Diagnosis IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE FIND DIAGNOSIS'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal; broker.Param[1].Value := txtDiagnosis.Text;

broker.Param[1].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE FIND FIRST ENCOUNTER

**Description:** This Remote Procedure Call (RPC) returns the internal entry number

of an OUTPATIENT ENCOUNTER file (#409.68) entry for the first

encounter for a patient in a specified date range.

**Related API:** 74 - SDOE FIND FIRST ENCOUNTER

**Parameter**

**Position Argument Type Required**

0 Patient ID Literal Yes

1 Begin Date/Time Literal Yes

2 End Date/Time Literal Yes

3 Search Flags Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE FIND FIRST ENCOUNTER; broker.Param[0].Value := txtPatient.Text;

broker.Param[0].PType := literal;

broker.Param[1].Value := txtBegin.Text;

broker.Param[1].PType := literal;

broker.Param[2].Value := txtEnd.Text;

broker.Param[2].PType := literal;

broker.Param[3].Value := txtFlags.Text;

broker.Param[3].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE FIND FIRST STANDALONE

**Description:** This Remote Procedure Call (RPC) returns the internal entry number

of an OUTPATIENT ENCOUNTER file (#409.68) entry for the first

standalone add/edit for a patient in a specified date range. Use same

date for begin and end dates for specific (single) date check. Standalone

encounter is an encounter with no parent and the originating process is

“Stop Code Addition”.

**Related API:** 72 - SDOE FIND FIRST STANDALONE

**Parameter**

**Position Argument Type Required**

0 Patient ID Literal Yes

1 Begin Date/Time Literal Yes

2 End Date/Time Literal Yes

3 Search Flags Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE FIND FIRST STANDALONE'; broker.Param[0].Value := txtPatient.Text; broker.Param[0].PType := literal; broker.Param[1].Value := txtBegin.Text;

broker.Param[1].PType := literal;

broker.Param[2].Value := txtEnd.Text;

broker.Param[2].PType := literal;

broker.Param[3].Value := txtFlags.Text;

broker.Param[3].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE FIND LAST STANDALONE

**Description:** This Remote Procedure Call (RPC) returns the internal entry number

of an OUTPATIENT ENCOUNTER file (#409.68) entry for the last

standalone add/edit for a patient in a specified date range. Standalone

encounter is an encounter with no parent and the originating process is

“Stop Code Addition”.

**Related API:** 75 - SDOE FIND LAST STANDALONE

**Parameter**

**Position Argument Type Required**

0 Patient ID Literal Yes

1 Begin Date/Time Literal Yes

2 Search Flags Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE FIND LAST STANDALONE'; broker.Param[0].Value := txtPatient.Text; broker.Param[0].PType := literal;

broker.Param[1].Value := txtBegin.Text;

broker.Param[1].PType := literal;

broker.Param[2].Value := txtFlags.Text;

broker.Param[2].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE FIND PROCEDURE

**Description:** This Remote Procedure Call (RPC) returns a boolean indicator on

whether a specific procedure is associated with an encounter.

**Related API:** 71 - SDOE FIND PROCEDURE

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

1 CPT IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE FIND PROCEDURE'; broker.Param[0].Value := txtEncounter.Text;

broker.Param[0].PType := literal;

broker.Param[1].Value := txtProcedure.Text;

broker.Param[1].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE FIND PROVIDER

**Description:** This Remote Procedure Call (RPC) returns a boolean indicator on

whether a specific provider is associated with an encounter.

**Related API:** 69 - SDOE FIND PROVIDER

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

1 Practitioner ID Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE FIND PROVIDER'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal;

broker.Param[1].Value := txtProvider.Text;

broker.Param[1].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE GET DIAGNOSES

**Description:** This Remote Procedure Call (RPC) returns an array of diagnoses for

an encounter.

NOTE: For encounters before 10/1/96, only scheduling data in the

OUTPATIENT DIAGNOSIS file (#409.43) may exist. It will only exist if the site required diagnoses as part of the check out process. This RPC will attempt to find this “old” data, reformat the data to meet the V POV structure and return the list of diagnoses as described in API# 56. (Only the diagnosis code internal entry number is available for “old” encounters.)

**Related API:** 56 - SDOE GET DIAGNOSES

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE GET DIAGNOSES';

broker.Param[0].Value := txtEncounter.Text;

broker.Param[0].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE GET GENERAL DATA

**Description:** This Remote Procedure Call (RPC) returns the zeroth and other

nodes of an outpatient encounter entry.

Format: <encounter node> ;; <node value>

NOTE: Currently (7/97) only the zeroth node is returned. Also, only

fields .01 thru .08 and .1 thru .13 of the zeroth node are returned. Other nodes and fields are not supported.

**Related API:** 76 - SDOE GET GENERAL DATA

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE GET GENERAL DATA'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal; broker.Call; memResults.Lines := broker.Results;end;

**Name:** SDOE GET PRIMARY DIAGNOSIS

**Description:** This Remote Procedure Call (RPC) returns the internal entry number of

the primary diagnosis code (^ICD9) for an encounter.

NOTE: For encounters before 10/1/96, this RPC will always return 0.

This primary diagnosis was not retrieved nor stored by the system for these “old” encounters.

**Related API:** 73 - SDOE GET PRIMARY DIAGNOSIS

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE GET PRIMARY DIAGNOSIS'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal; broker.Call; memResults.Lines := broker.Results;

end;

**Name:** SDOE GET PROCEDURES

**Description:** This Remote Procedure Call (RPC) returns a subscripted array of

CPTs for an encounter.

NOTE: For encounters before 10/1/96, only scheduling data in the

SCHEDULING VISITS file (#409.5) may exist. It will only exist if the site required procedures as part of the check out process. This RPC will attempt to find this “old” data, reformat the data to meet the V CPT structure and return the list of procedures as described in API# 61. (Only the CPT code internal entry number and count are available for “old” encounters.)

**Related API:** 61 - SDOE GET PROCEDURES

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE GET PROCEDURES'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal; broker.Call; memResults.Lines := broker.Results;

end;

**Name:** SDOE GET PROVIDERS

**Description:** This Remote Procedure Call (RPC) returns a subscripted array of

providers for an encounter.

NOTE: For encounters before 10/1/96, only scheduling data in the

OUTPATIENT PROVIDER file (#409.44) may exist. It will only exist if the site required provider as part of the check out process. This RPC will attempt to find this “old” data, reformat the data to meet the V PROVIDER structure and return the list of providers as described in API# 58. (Only the provider internal entry number is available for “old” encounters.)

**Related API:** 58 - SDOE GET PROVIDERS

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE GET PROVIDERS'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal; broker.Call; memResults.Lines := broker.Results;end;

**Name:** SDOE GET ZERO NODE

**Description:** This Remote Procedure Call (RPC) returns the zeroth node of an

outpatient encounter.

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the

zeroth node are returned.

**Related API:** 98 - SDOE GET ZERO NODE

**Parameter**

**Position Argument Type Required**

0 Encounter IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE GET ZERO NODE'; broker.Param[0].Value := txtEncounter.Text; broker.Param[0].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE LIST ENCOUNTERS FOR DATES

**Description:** This Remote Procedure Call (RPC) returns a list of outpatient

encounters for a date range.

Format: <encounter ien> ;; <zeroth node of encounter>

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the

zeroth node are returned.

**Related API:** 99 - SDQ SCAN

**Parameter**

**Position Argument Type Required**

0 Begin Date/Time Literal Yes

1 End Date/Time Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE LIST ENCOUNTERS FOR DATES';

broker.Param[0].Value := txtBegin.Text;

broker.Param[0].PType := literal;

broker.Param[1].Value := txtEnd.Text;

broker.Param[1].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE LIST ENCOUNTERS FOR PAT

**Description:** This Remote Procedure Call (RPC) returns a list of outpatient

encounters for a specified patient and specified date range.

Format: <encounter ien> ;; <zeroth node of encounter>

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the

zeroth node are returned.

**Related API:** 99 - SDQ SCAN

**Parameter**

**Position Argument Type Required**

0 Patient ID Literal Yes

1 Begin Date/Time Literal Yes

2 End Date/Time Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE LIST ENCOUNTERS FOR PAT'; broker.Param[0].Value := txtPatient.Text;

broker.Param[0].PType := literal;

broker.Param[1].Value := txtBegin.Text;

broker.Param[1].PType := literal;

broker.Param[2].Value := txtEnd.Text;

broker.Param[2].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE LIST ENCOUNTERS FOR VISIT

**Description:** This Remote Procedure Call (RPC) returns a list of outpatient

encounters for a specified visit.

Format: <encounter ien> ;; <zeroth node of encounter>

NOTE: Currently (7/97) only fields .01 thru .08 and .1 thru .13 of the

zeroth node are returned.

**Related API:** 99 - SDQ SCAN

**Parameter**

**Position Argument Type Required**

0 Visit IEN Literal Yes

**Delphi**

**Example:** begin

memResults.Lines.Clear; broker.RemoteProcedure := 'SDOE LIST ENCOUNTERS FOR VISIT'; broker.Param[0].Value := txtVisit.Text; broker.Param[0].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

**Name:** SDOE PARSE GENERAL DATA

**Description:** This Remote Procedure Call (RPC) will parse the data returned by

the “SDOE GET GENERAL DATA” RPC into individual field nodes.

Format: <encounter field #> ;; <field value>

**Related API:** 78 - SDOE PARSE GENERAL DATA

**Parameter**

**Position Argument Type Required**

0 Encounter Data List Yes

1 Encounter Parse Format Literal Yes

**Delphi**

**Example:** begin

// get data needed for the 'parse' RPC

memResults.Lines.Clear;

broker.RemoteProcedure := 'SDOE GET GENERAL DATA';

broker.Param[0].Value := txtEncounter.Text;

broker.Param[0].PType := literal;

broker.Call;

// call parser RPC using data retrieved from above

broker.RemoteProcedure := 'SDOE PARSE GENERAL DATA';

broker.Param[0].Value := '.SDY';

broker.Param[0].PType := list;

broker.Param[0].Mult['0'] := Piece(broker.Results[0],';;',2);

broker.Param[1].Value := txtFormat.Text;

broker.Param[1].PType := literal;

broker.Call;

memResults.Lines := broker.Results;

end;

Error Processing

**Structure**

Most of the APIs defined in the ACRP Interface Toolkit have error messages that may be passed back to the calling application in the event that an error does occur. The structure and design of these error messages is the same as that used by the VA FileMan Database Server APIs.

The structure is based on the DIALOG file (# .84) and supporting utilities. For detailed discussion on the structure and utilities, see the following in the VA FileMan Programmer Manual:

|  |  |
| --- | --- |
| **Section** | **Subsection** |
| Database Server | How the FileMan Database Server Communicates |
| Database Server | MSG^DIALOG()  This is the utility API that a developer can use to help process returning error messages. |
| Database Server | BLD^DIALOG() |
| Database Server | $$EZBLD^DIALOG() |

**Error Arrays**

As described in the VA FileMan documentation, error messages are stored, by default, in the ^TMP(“DIERR”,$J) array. However, a developer can explicitly indicate the name of an array where all error messages should be placed. This array name is passed as a parameter as the following example shows:

S X=$$GETOE^SDOE(OE,”MYERROR”)

If any errors occurred during this function call, the errors will be placed in the MYERROR array.

If the following call was made instead…

S X=$$GETOE^SDOE(OE)

…then any errors would be placed as nodes in ^TMP(“DIERR”,$J) ARRAY.

Whenever the developer makes an AIT call and does not explicitly indicate an error message array, any nodes in the ^TMP(“DIERR”,$J) array are killed. As a result, the developer will know that any data that exists after the call was generated by that call. If an array is explicitly specified, then it is the responsibility of the developer to manage that array as required by the application.

Finally, in order to make sure that the arrays are deleted before the application quits, use the following call that deletes the arrays and related variables:

DO CLEAN^DILF

If the developer has instructed the AIT call to place error messages into a specific array, the developer must be sure to clean up that array.

For documentation on CLEAN^DILF, see the following in the VA FileMan Programmer Manual:

|  |  |
| --- | --- |
| **Section** | **Subsection** |
| Database Server | CLEAN^DILF |

**AIT Error Processing Tools**

**Checking For Errors**

Many times when making AIT API calls, the developer will have already determined that parameters being passed are valid, such as passing in a date range or a patient internal entry number. As a result, there is no need to look for errors.

However, in the event there is a possibility that an error may be encountered by the AIT API, the developer can use the $$ERRCHK^SDQUT call to easily determine if an error has occurred, as the following example shows:

EN N QUERY

D OPEN^SDQ(.QUERY)

IF '$$ERRCHK^SDQUT() D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

IF '$$ERRCHK^SDQUT() D PAT^SDQ(.QUERY,101,"SET")

IF '$$ERRCHK^SDQUT() D DATE^SDQ(.QUERY,2970101,2971231,"SET")

IF '$$ERRCHK^SDQUT() D ACTIVE^SDQ(.QUERY,"TRUE","SET")

IF '$$ERRCHK^SDQUT() D LAST^SDQ(.QUERY)

IF '$$ERRCHK^SDQUT() F Q:$$BOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

. D PRIOR^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

D CLEAN^DILF

Q

For more information on $$ERRCHK^SDQUT, see its API definition in this manual.

**Debugging**

Another tool available to the developer and Customer Service staff is the SDEBUG variable. If this variable is defined, the AIT will automatically display any errors.

For example, if there is an error in the date range specified for a call to DATE^SDQ(), then the error will be displayed, as the following shows:

Code: EN N QUERY

***S SDEBUG=””***

D OPEN^SDQ(.QUERY)

D INDEX^SDQ(.QUERY,"PATIENT/DATE","SET")

D PAT^SDQ(.QUERY,101,"SET")

***D DATE^SDQ(.QUERY,2980101,2971201,"SET")***

D ACTIVE^SDQ(.QUERY,"TRUE","SET")

D LAST^SDQ(.QUERY)

F Q:$$BOF^SDQ(.QUERY) D

. D PROCESS(.QUERY)

. D PRIOR^SDQ(.QUERY)

D CLOSE^SDQ(.QUERY)

D CLEAN^DILF

***K SDEBUG***

Q

Error Display: Error Number: 4096800.022

Date range is not valid.

Date Range: '2980101' to '2971201'.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

As shown in the example, the developer can temporarily set the SDEBUG variable directly in the routine during testing. Customer Service personnel could do the same or as part of an ENTRY ACTION for an Option or Protocol.

Appendix - Argument Definitions

**Name: Action**

Description: This parameter indicates whether a property should be set or

retrieved.

Valid Values

GET retrieve property

SET set property

Data Type: String

Formal List Variable: action

**Name: Begin Date/Time**

Description: Beginning Date and time.

Format: VA FileMan

Time: Optional

Data Type: Date/Time

Formal List Variable: begin\_date

**Name: CPT IEN**

Description: This is the internal entry number of an entry in the CPT

[#81 - ^ICPT] file.

Data Type: Pointer

Formal List Variable: cpt**Name: Diagnosis IEN**

Description: This is the internal entry number of an entry in the ICD

DIAGNOSIS [#80 - ^ICD9] file.

Data Type: Pointer

Formal List Variable: diagnosis

**Name: Direction [optional]**

Description: This parameter indicates whether a search should start at

the beginning of the list and work down or start at the end of

the list and work up.

Valid Values

FORWARD start at beginning and work down

BACKWARD start at end and work up

Data Type: String

Formal List Variable: scan\_direction

**Name: Encounter Data**

Description: This array contains subscripts that correspond to each node

of data for an outpatient encounter entry.

NOTE: Currently (7/97) only the zeroth node is returned.

Also, only fields .01 thru .08 and .1 thru .13 of the zeroth are returned. Other nodes and fields are not supported.

For detail information regarding the fields, see the

data dictionary for the Outpatient Encounter file (#409.68).

Data Type: Array Reference

Formal List Variable: encounter\_data**Name: Encounter IEN**

Description: This is the internal entry number of an entry in the

OUTPATIENT ENCOUNTER [#409.68 - ^SCE] file.

Data Type: Pointer

Formal List Variable: encounter

**Name: Encounter Parse Format**

Description: Defines format for parsed data.

Valid Values

INTERNAL use internal format

EXTERNAL external/display format

Data Type: String

Formal List Variable: format**Name: Encounter Parsed Data**

Description: This parameter will return parsed encounter data.

The format will be based on the Encounter Parse Format

parameter, internal or external.

Each field will have its data returned in the array using the

field number as the subscript, i.e., SDY(.01)=2970712.13.

NOTE: Currently (7/97) only the zeroth node is returned.

Also, only fields .01 thru .08 and .1 thru .13 of the zeroth are returned. Other nodes and fields are not supported.

For detail information regarding the fields, see the data dictionary for the Outpatient Encounter file (#409.68).

Data Type: Array Reference

Formal List Variable: parsed\_data

**Name: Encounter Query Active Status**

Description: This parameter indicates the status of the query object.

Valid Values

1 query active

0 query not active

Data Type: Boolean

Formal List Variable: status**Name: Encounter Query Filter**

Description: This parameter can be used by the developer to screen out

entries using M code. The code must set $T.

The following information is available to the developer when

this logic is executed:

Variable Description

Y Internal Encounter Entry Number

Y0 Zeroth node of encounter entry

NOTE: Depending on the file, not all fields of the zeroth

node may be returned. Only those fields supported by the application will be returned. Non-supported fields will be null.

NOTE: This filter logic should only return a boolean value

via $TEST. Developers should not set application variables as part of this filter logic.

Data Type: String

Formal List Variable: filter

**Name: Encounter Query Handle**

Description: The query handle obtained as a result of a call to the SDQ

OPEN method of the query object.

More than one query object instance can be created by a

process. This handle indicates which instance to use when

various query object methods are called.

Data Type: String

Formal List Variable: query**Name: Encounter Query Index**

Description: This parameter is used to indicate which cross reference the

query should use when producing the result set.

Valid Values

PATIENT use patient cross reference

PATIENT/DATE use patient by encounter date cross

reference

DATE/TIME use the encounter date cross reference

VISIT use the visit cross reference

Data Type: String

Formal List Variable: index

**Name: End Date/Time**

Description: Ending date and time.

Format: VA FileMan

Time: Optional

Data Type: Date/Time

Formal List Variable: end\_date**Name: Error Array [optional]**

Description: Literal identifying the array where error information should

be stored. The calling application can then process any

errors when control is returned.

The error array is in DIALOG utility format.

Data Type: Array Reference

Formal List Variable: errors

**Name: List of V CPT Entries**

Description: An array of V CPT (#9000010.18) entries for a visit. The

array is subscripted by the V CPT internal entry numbers.

Each subscript entry value equals the zeroth node of the V

CPT entry. See the data dictionary for more information on

the value of each piece.

The top level, non-subscripted value is a count of the number

of entries found.

Data Type: Array Reference

Formal List Variable: cpt\_list**Name: List of V POV Entries**

Description: An array of V POV (#9000010.07) entries for a visit. The

array is subscripted by the V POV internal entry numbers.

Each subscript entry value equals the zeroth node of the V

POV entry. See the data dictionary for more information on

the value of each piece.

The top level, non-subscripted value is a count of the number

of entries found.

Data Type: Array Reference

Formal List Variable: dx\_list

**Name: List of V PROVIDER Entries**

Description: An array of V PROVIDER (#9000010.06) entries for a visit.

The array is subscripted by the V PROVIDER internal entry

numbers.

Each subscript entry value equals the zeroth node of the V

PROVIDER entry. See the data dictionary for more

information on the value of each piece.

The top level, non-subscripted value is a count of the number

of entries found.

Data Type: Array Reference

Formal List Variable: provider\_list

**Name: Patient ID**

Description: This is the internal entry number of an entry in the

PATIENT [#2 - ^DPT] file.

Data Type: Pointer

Formal List Variable: dfn**Name: Practitioner ID**

Description: This is the internal entry number of an entry in the NEW

PERSON [#200 - ^VA(200)] file for a practitioner.

Data Type: Pointer

Formal List Variable: provider

**Name: Scan Callback Logic**

Description: Application M code executed by the query object SDQ SCAN

method for each entry found in the record scan process.

The following variables are available at the time this

callback code is executed:

Variable Description

Y Internal Encounter Entry Number

Y0 Zeroth node of encounter entry

SDSTOP Set to 1 to tell Scan to stop processing

The callback code should look similar to either of the

following:

>> Parameter Passing Approach:

D CALLBACK^XX(Y,Y0,.SDSTOP)

NOTE: If this approach is used, then SDSTOP must be

passed by reference.

>> Classic M Approach

D CALLBACK^XX

Data Type: String

Formal List Variable: callback**Name: Search Flags**

Description: This parameter allows developers to set specific flags that

are used as an API searches encounter records. The flags

indicate how the API should function.

Character Description

C Use only completed encounters

Data Type: String

Formal List Variable: flags

**Name: Visit IEN**

Description: This is the internal entry number of an entry in the VISIT

[#9000010 - ^AUPNVSIT] file.

Data Type: String

Formal List Variable: visit