System Concepts Guide

Release 5.0

iLock Security Services
System Concepts Guide

Subject

Basic concepts and operation of the iLock™ Security Services.

Software Supported

iLock Security Services

- iLock Enterprise 5.0
- jLock 5.0
- orb2 for Java Security Services 5.0
- webLock 5.0
- c/Lock 5.0

Revision History

Product Release	September 2002
Release 4.1	March 2003
Release 4.2	February 2004
Release 4.3	October 2004
Release 4.4	September 2006
Release 5.0	December 2007

2AB, Inc. disclaims the implied warranties of merchantability and fitness for a particular purpose and makes no express warranties except as may be stated in its written agreement with and for its customer. In no event is 2AB, Inc. liable to anyone for any indirect, special or consequential damages.

The information and specifications in this document are subject to change without notice. Consult your 2AB, Inc. marketing representative for product or service availability.

U.S. Government Restricted Rights. The Software Program(s) and Documentation furnished under this Agreement were developed at private expense and are provided with Restricted Rights. Any use, duplication, or disclosure by and for any agency of the U.S. Government shall be subject to the Restricted Rights applicable to commercial computer software under FAR Clause 52.227-19 or DFAR Clause 252.277-7013 or any successor thereof.

Trademarks

The 2AB logo, iLock, the iLock logo and orbLock are registered trademarks of 2AB, Inc. 2AB, Inc., c/Lock, eXplorer, iLock Security Services, jLock, orb2, webLock and Xcon are trademarks of 2AB, Inc.

Microsoft, Windows, Windows NT, Windows 2000 and Windows XP are registered trademarks of the Microsoft Corporation.

OMG, Object Management Group, OMG Interface Definition Language (IDL), Unified Modeling Language and UML are trademarks of the Object Management Group. CORBA and IIOP are registered trademarks of the Object Management Group.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

HP-UX is a registered trademarks of the Hewlett-Packard Company.

Apache and Tomcat are trademarks of The Apache Software Foundation.

AIX and Domino are trademarks of the International Business Machines Corporation in the United States or other countries or both.

iPlanet, J2EE, J2SE, Java, JavaScript, JavaServer Pages, JKD, Solaris, Sun and Sun Microsystems are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

All other brand or product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

Figures

About This Document

Who should read this guide?

Programmers and administrators who are securing applications using the iLock Security Services.

Technical Support

Your feedback is important to us. Please provide input to the 2AB Technical Support Staff.

You can communicate comments to and request help from the Technical Support Staff via the following methods:

Telephone

U.S. or Canada
877.334.9572 (Toll Free)

All Other Countries
+1.205.621.7455

Email

support@2ab.com

Chapter 1 Security Overview

With the explosion of online services, controlling access to enterprise computing resources is increasingly critical. Today, one only has to open a newspaper or turn on the television to understand that there is growing concern regarding the privacy of personal and business information. This concern is coming from both individuals and businesses, and government regulations are increasingly addressing these concerns. Although computer systems have traditionally been "secured," it is evident that future systems will be expected to enforce a higher quality and a finer granularity of security mechanisms.

One important aspect of securing future systems is the ability to control, with a fine level of granularity, the ability of individuals and systems to access information and functionality that systems provide. 2AB, Inc. has introduced the iLock Security Services software to address this problem.

Securing distributed computing systems typically requires that security software can perform one or more of the following functions:

- Authentication
- Message Protection
- Access Control
- Auditing
- Administration

1.1 Authentication

Authentication is the term used to indicate that an individual or system has proven its identity. In a distributed computing environment, authentication may work in both directions. That is, a client may be required to prove its identity to some service, and conversely, some service might be required to prove its identity to its client. The authentication function is the cornerstone for all other security functions. Without reliable authentication, all other security features that may be used are effectively meaningless.

There are various techniques used to authenticate individuals and/or systems. They include:

- Shared Secret - User ID and Password
- Physical Tokens, such as ATM cards, Smart cards...
- Digital Certificates
- Biometrics, such as retina scans, thumbprint readers…

1.2 Message Protection

Message protection is the term used to indicate that messages (typically in transit across networks) are protected from being viewed or modified by unauthorized persons. To protect messages from being viewed, the sending party typically encrypts messages, and the receiving party has the ability to decrypt the message. To protect the message from being modified, there is typically encrypted information about the message (perhaps a hash of the message) that can only be

created by the sender and read by the receiver. If this information does not match the message contents, there is a strong indication that the message has been tampered with.

Message protection techniques should almost always be used when messages travel through public networks (e.g. the Internet). Enterprises may or may not perceive the need for message protection within internal private networks.

1.3 Access Control

Access control is the term used to describe mechanisms that ensure that authenticated individuals and/or systems can only access protected resources for which they have permission. The term *protected resource* is used to indicate anything that needs protecting. It could be a computer system, some specific functionality of a system or some specific information that a system manages.

There are two aspects to access control.

1. **Access Decision** - is the aspect of access control that makes the decision to allow access or not. This decision is based on the security policy that has been associated with the protected resource.
2. **Access Enforcement** - is the aspect of access control that enforces the result of the access decision.

A system may have requirements to control access to a wide variety of protected resources. Some of these resources may be known to infrastructure software, such as operating systems and/or communications middleware. Other resources may only be known to business logic software. This may be information such as a customer's credit history.

1.3.1 Infrastructure Access Control

Infrastructure software is software used by business software developers that handle functions common to a wide range of business functions. This would include operating systems, database management systems, communications middleware and so forth. The access control that can be placed on resources known to infrastructure software is known as Infrastructure-Based Access Control.

There are many examples of resources that are typically protected by infrastructure software. Operating systems typically protect files using **read**, **write** and **execute** permissions. Database management systems typically protect access to databases and database tables. Communication middleware typically protects invocation of remote procedure calls, operations or methods.

The infrastructure software is responsible for both the access decision and the enforcement aspects of access decision. Obviously, the infrastructure can use third party access control software to assist in the decision/enforcement process; however, it is ultimately the infrastructure software's responsibility. Often, there are hooks that can be used by third party software to protect specific resources. For example, there are numerous products that specialize in protecting access to Web pages.

1.3.2 Application Layer Access Control

There are resources that need to be protected that only business logic can understand. For example, a system might have customer records that include general information, billing information, transaction history and credit history. Obviously, no infrastructure software will understand these resources; however, it might be necessary to control who is allowed to do what with each of these resources. It is the responsibility of the business logic to control the access to these

resources. In fact, with the increasing requirements for privacy of personal information, this is becoming a very important component of the overall access control requirements.

Application systems have traditionally implemented access control for business logic resources within the code of the business application. At first, this appears to be a rather easy solution; however, the fact that security policies are embedded in application code can significantly increase the long-term maintenance costs for the system. A simple change to a security policy can result in all of the costs associated with deploying a new release. Some systems may write code to make the security policy configurable; however, this typically results in different administrative procedures for different systems.

When security policy is embedded in source code, it is impossible to determine the security policy without reviewing the source code to locate the specific access control logic. This makes it difficult for an auditor to determine what the policy is and whether the application software is performing authorization appropriately to meet business and legislative security policies. It is important to maintain security policies outside of source code, in a format that can be easily reviewed, to facilitate an audit process. Should problems be discovered during a review/audit, the policies can be changed without disruption to the deployed business software.

Another issue, related to embedding access decision logic within the business logic, is auditing. Each application must provide a means for creating audit records and a means for security administrators to view/analyze those audit records. This forces business logic developers into the business of developing infrastructure software. In addition, there is the real possibility that different applications systems will implement different auditing strategies and tools, leading to additional training and overhead for security administrators.

1.4 Auditing

Auditing is the term used to describe the recording of information needed to detect and investigate security violations. The auditing mechanism must provide the necessary tools to view and examine the recorded information.

1.5 Administration

Administration is the term used to describe the management of security information required to secure access to computing resource. This information consists of information about valid users, resources to be secured and the rules that govern access to secured resources.

Security products provide administrative functionality in the form of programs that are to be used by an enterprise's security administrators. There may also be APIs provided that allow an enterprise to build specialized administrative tools. These APIs can also provide secured applications the ability to dynamically define security information that governs the secure access to its resources.

Chapter 2 iLock Security Services Overview

iLock Security Services products are a collection of software components that an enterprise can use to secure access to a wide variety of computing resources that belong to that enterprise. The types of enterprise resources that can be protected include distributed applications, database elements, Web pages and so forth. In fact, any enterprise resource that can be assigned a unique name can be secured, and these resources can exist in various infrastructure environments such as Web Servers, J2EE Application Servers and CORBA-based systems.

The components that make up the iLock Security Services include distributed services that help manage access to secured resources, administrative tools that manage these distributed services and programming interfaces (APIs) that can be used by applications to control access to secure enterprise resources.

2.1 Security Center

The Security Center serves as a repository of information related to securing enterprise computing resources and processes queries from iLock Components that need the security-related data it manages. This includes information about users, secured resources and security policies. The Security Center is the central component of the iLock Security Services software.

An enterprise may choose to run multiple instances of the Security Center. Each instance manages a unique set of security related information about users, resource and policies. Each instance must be assigned a unique name referred to as the *instance name*. If more than one instance is run on the same machine, each instance must be assigned a unique network port address. The instance name and port may be specified when starting the Security Center.

2.2 iLock Components

The software components that actually control access to secured resources are referred to as iLock Components. Each iLock Component is a collection of software components designed to protect resources unique to a particular computing environment. iLock Components use the Security Center as the repository for security-related configurations and policies. iLock Security Services currently supports four iLock Components:

1. jLock - provides administration, authentication and access control facilities for applications running on J2EE Application Servers, servlets or standalone Java applications. Basic, Standard and Enterprise editions of jLock are available that provide levels of functionality that range from basic to advanced. For more information, please see the *iLock Security Services: jLock User Guide.*
2. webLock - secures access to Web Resources such as Web pages, servlets and JavaServer Pages. For more information, please see the *iLock Security Services: webLock User Guide.*
3. orbLock - secures access to resources in an OMG CORBA-based environment. For more information, please see the *iLock Security Services: orbLock User Guide.*

2.3 Locating Security Service Instances

Multiple instances of the Security Center and iLock Components can run on multiple processors located throughout a network. These components must have a way of locating each other in a distributed environment. This configuration is done using a special tool called **iconfig**.

The **iconfig** utility should be run on all machines that will run applications that use one of the iLock Components and on all machine where administrative tools will be used. This utility is used to configure the host and port information for all possible Security Center instances.

Chapter 3 Security Center

The Security Center is a distributed service that manages persistent information relating to securing enterprise resources. These include resource definitions, security policies, user definitions and so forth.

3.1 Security Center Instances

A single running instance of the Security Center is referred to as a Security Center Instance. An enterprise may run one or more copies of the Security Center. Each instance of the Security Center represents a domain of security information that is available to the iLock Component software that uses that instance of the Security Center.

Many enterprises will only run a single instance of the Security Center. Larger, more complex enterprises may choose to run multiples instances of the Security Center, with each managing a different domain of security-related information. When using an iLock Component, one specifies the Security Center instance it will use.

Example An enterprise might have separate Security Center instances to manage separate organizational departments. An instance of the jLock Component might specify that it will use the Security Center named *accounting*, and a different use of the jLock Component might specify that it will use the Security Center named *engineering*.

On the other hand, an enterprise might manage the security of the entire enterprise with a single instance. One or more components will interact with a Security Center instance to provide security services for a particular infrastructure environment.

3.2 Objects Managed by Security Center

The Security Center manages the information required to secure enterprise resources. This section describes different forms of security-related data managed by the Security Center.

3.2.1 Users

Users are defined to represent either a human user of systems or a system itself. For example, an accounting system might be defined as user *Acct101*, and a human user of that system might be defined as *Susan*. The following information can be specified when defining a user:

- First Name
- Middle Name or Initial
- Last Name
- User ID
- Optional Data
- User Domain

The first, middle and last names are used to identify the human or system. The user ID is used when authenticating a user (i.e. logging on). The User ID must be unique within the Security Center. The optional data field is a string that can carry any desired value (e.g. a users email address). The user's domain is simply a group of users. Controlling management of different user domains is a planned enhancement for a future release.

Users are associated with zero or more security attributes. iLock Components that enforce access control rules use these attributes to make the access decisions.

The Security Center manages User IDs and Passwords used by iLock Components that perform user authentication with User IDs and Passwords. The jLock Component supports user authentication with either the JII or JAAS Components. The orbLock Component supports authentication with both the CSIv2 and CSS Components.

User IDs may also be managed to provide a mapping to security attributes (Access IDs, Groups and Roles) that might be used in systems that perform authentication but have no security attribute mapping facility (J2EE application servers, for instance).

3.2.2 Security Attributes

Security attributes, used to define an identity or a privilege, are associated with user IDs to provide a set of privileges for that user and will be used in security policies (rules) to control access to secured resources. The Security Center supports three types of security attributes:

1. Access ID - an attribute that denotes an ID that can be used to access certain secured resources. Each individual user might be associated with a unique Access ID, or multiple users might be associated with the same Access ID. The Security Center always defines a special Access ID that has the value of *Public*. This special Access ID can be used in security policies to indicate that anyone, even users that are not assigned an Access ID, will be allowed to access a given resource.
2. Group - an attribute that denotes an ID that can be assigned to one or more users, typically to indicate that all members of a given group possess certain privileges. The Group attribute is typically associated with a group of users that are related by geography, enterprise, department, etc.
3. Role - an attribute that denotes the role (typically the job function) that users possess. The same Role attribute can be assigned to many users. The Role attribute is somewhat different in that it can also be associated with a Group attribute as well as a user definition. If a group has one or more associated roles, then any users that are associated with that group are automatically associated with the roles possessed by that group

Note It is important to note that security attributes, not user definitions, are used to define security policies. User definition may be associated with one or more security attributes.

The security attribute is a data structure that is composed of three data members.

1. Attribute Type - Access ID, Group or Role.
1. Defining Authority - specifies the entity (organization) that has defined and certified the attribute. Some enterprises may only define a single defining authority, while others may define multiple defining authorities.
2. Value - the value for an attribute will typically be a user name, group name or role name. For example, a Group attribute might have a value of *Engineering*.

There is a special security attribute that is automatically created and managed by the Security Center. It is an Access ID whose value is *Public* and whose defining authority is *OMG*. Some iLock Components will always assign every user to that attribute, even unauthenticated users. Policies can then be built that will allow access to all users.

3.2.3 Secured Resources

A secured resource is anything within an enterprise computing environment that should be protected from unauthorized access. That might be a processor, an application, a particular function of an application, a file, a database or a particular record in a database.

The types of secured resources currently supported by the Security Center are Basic Resources, JAAS Resources, Web Resources, RAD Resources, and CORBA Operations. The meaning of the resource will depend on the iLock Component protecting the resource. For example, a RAD Resource can represent anything that an application wants to protect (such as a database record), whereas a CORBA Operation is intended to represent a single, CORBA IDL-defined operation.

Although each of these secured resources are represented in the same format internally, each is presented visually to the administrator using somewhat different formats. The Security Center presents different views of resources depending on the context in which they will be used. This effectively means that an administrator will view different types of resources with views that present different formats. Secured resources are identified by a unique name within the Security Center. Although different iLock Components will control access to secured resources that may be identified in different ways, all secured resources are eventually mapped to a common format within the Security Center.

Secured resource may be associated with zero or more security policies (see Section 3.2.4, **Security Policies**). When an iLock Component is making an access decision for a resource, the user accessing that resource must satisfy each of the security policies that are associated with that resource.

3.2.4 Security Policies

A Security Policy consists of the rules that govern access to secured resources. Security policies are exposed in one of two ways. The first and most common way is referred to a "Resource Policy." This is a single unnamed policy that is associated with one and only one secured resource. The second way is referred to as a "Policy Group." Policy groups are used where multiple resources can share the same policy. The policy group has a name, a policy and a list of resources that the policy applies to. Resources may be assigned to multiple policy groups.

A secured resource may have multiple operations that people may want to perform on that resource (e.g read, write, ..). A Security Policy is comprised of a set of operation policies that govern whether or not a specific operation should be allowed on a particular resource. For example, if a specified file is the resource to be protected, there will be different rules that govern the **read** and **write** operations.

Since a security policy must be capable of different rules for different operations, the security policy is comprised of one or more *operation policies*. For example, a security policy might consist of two operation policies, one for the **read** operation and another for the **write** operation. Any operation can be defined. For example, we could define an operation named **obliterate**.

Each operation policy will consist of one or more rules that define how the access decision will be made. Since these rules can be time dependent, we call these *Timed Rules*.

A Timed Rule is a rule used to make access decisions and consists of a rule type, a collection of security attributes, entitlement rules and time attributes that indicate the time constraints to place on the rule.

Timed Rule Types

There are five types of Timed Rules:

1. Required Attributes

   Indicates that a user must possess all of the attributes defined in the rule in order to meet the access requirements of the rule. For example, if a Timed Rule defines two attributes, *AccessID-Smith* and *Role-Doctor*, then a user is required to have both of these attributes to meet the access requirements.

2. Any Attributes

   Indicates that if a user possesses any one of the attributes defined in the rule, the user then meets the access requirements of the rule. For example, if a Timed Rule defines two attributes, *AccessID–Smith* and *Role-Doctor*, then a user having either of these two attributes will meet the access requirements.

3. Deny Attributes

   Indicates that if a user possesses any of the attributes defined in the rule, the user will be denied access. For example, if a Timed Rule defines two attributes, *AccessID–Smith* and *Role-Doctor*, then a user having either of these attributes will be denied access.

4. Anybody Allowed

   Indicates that all users, regardless of their attributes, will meet the access requirements.

5. Nobody Allowed

   Indicates that no users, regardless of their attributes, will meet the access requirements.

Precedence of Timed Rule Types

When evaluating a sequence of Timed Rules, the Timed Rule types are evaluated and enforced in the following precedence order:

- Nobody Allowed
- Deny Attributes
- Required Attributes
- Any Attributes
- Anybody Allowed

For example, consider a sequence of rules that has a rule type of Nobody Allowed and another one that has a type of Any Attributes. The rule of Nobody Allowed will take precedence, and even people having attributes that would match the Any Attributes rule will be denied. This precedence is time-dependent. If a higher precedence rule type does not meet a specified time constraint, its precedence is of no importance. For example, if the Nobody Allowed type is only to be enforced on Tuesdays, it precedence is meaningless on Wednesday.

Timed Rule Security Attributes

A Timed Rule can have a sequence of Security Attributes that are used to make access decisions. The security attributes are required for the Required Attributes, Any Attributes and Deny Attributes types. They have no meaning for the Anybody Allowed and Nobody Allowed rule types.

Timed Rule Entitlement Rules

A Timed Rule can have a collection of Entitlement Rules that are enforced when making access decisions. An entitlement rule consists of a variable name, a variable value, a variable type and a variable relationship (e.g. greater than). When an application requests an access decision, it must provide the variable name and a value. The access decision engine will validate the variable's type and ensure that the relationship specified in the entitlement rule is correct. For example, an entitlement rule might have a variable named "Salary," a value of "40000," a type of decimal, and a relationship of "greater than." If an application provides entitlement data with a variable named "Salary" and a value of "39000," the entitlement rule will deny access. On the other hand, if the value is "410000," the entitlement rule will allow access.

Timed Rule Time Constraints

Each Timed Rule can specify the time frame during which the rule is to be effective. The time constraint may be in the form of an exact time (e.g. 10:01 AM January 23, 1999) or in the form of a recurring-day-of-the-week. A recurring-day-of-the-week time constraint may be used to make a rule effective Tuesday through Friday of every week.

Combining Timed Rules

Once you have an understanding of the different rule types and their precedence, it becomes quite easy to build security policies, consisting of multiple rules, which would otherwise seem difficult to express.

As an example, suppose you want to allow everyone in the Engineering Department to have access to a set of documents (secured resources); however, the exception to this rule is that Lawyers in the Engineering Department should not be allowed to access the documents. To accomplish this, simply create two rules. The first rule, an Any Attributes type, allows access to anybody with a group attribute of *engineering*. The second rule, a Deny Attributes type, prevents anyone with a role attribute of *lawyer* from accessing the documents.

3.3 Object Type Relationships

Figure 3.1, **Object Type Relationships**, illustrates the relationship between the various object types managed by the Security Center.

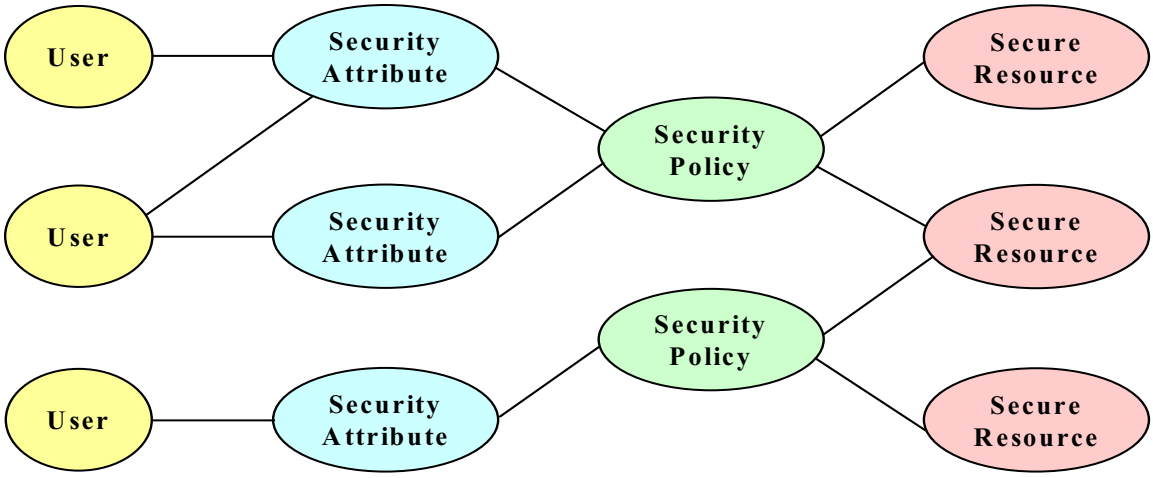


Figure 3.1 Object Type Relationships

In the above illustration, note that the solid lines indicate a relationship between two data elements. Some important points to note about these relationships are:

- A user can be associated with zero or more security attributes.

- Different users can be associated with the same security attribute.
- A security policy can be associated with zero or more security attributes.
- Different security policies can be associated with the same security attribute.
- Secured resources can be protected by one or more security policies.
- Different secured resources can share the same security policy.

3.4 LDAP and Custom Interfaces

Many enterprises have existing repositories that maintain information about users, user IDs, and security attributes (groups and roles) that are associated with these users. A common infrastructure for managing these repositories is an LDAP-based (Lightweight Directory Access Protocol) directory service. Other enterprises manage this task with other infrastructures, including those developed within the enterprise.

The Security Center provides the ability to interface with these infrastructures to retrieve information about users and their associated security attributes. A special LDAP option allows the Security Center to interface with an existing LDAP based service. Implementation of the **CustomUserManager** interface will allow those enterprises to have the Security Center interface with non-LDAP-based infrastructures.

3.5 Security Center Administration

There are two tools provided to allow administrators to manage the Security Center. There is a program with a graphical user interface that allows administrators to manage resources, policies, users and their associated attributes. Similarly, there is a command-line tool that can perform the same tasks in a batch mode by reading management tasks to be performed from a text file.

The Security Center Administration Tool is a graphical user interface that is provided to manage Security Center instances. This tool can connect to any instance of a Security Center on the network and manage its resources. It can manage security policies, secured resource definitions and user/privilege information and can support multiple types of secured resources appropriate for different computing environments and different iLock Components.

The Security Center Batch Administration Tool, just like the GUI-based Administration Tool, provides security administrators with the ability to define and manage the security information within the Security Center. This tool uses a text-based file that provides instructions that describe the administrative tasks to be performed. This tool can connect to different instances of the Security Center.

Chapter 4 iLock Components

The software components that actually control access to secured resources are referred to as iLock Components. Each iLock Component is a collection of software components designed to protect resources unique to a particular computing environment. iLock Components communicate with an instance of the Security Center to retrieve information about security policies, secured resource and users. Since this communication can traverse a network, it is important to optimize the communication between iLock Components and the Security Center.

All iLock Components provide caching capabilities to minimize the required communication with a Security Center instance. The Security Center will notify iLock Components when persistent security information has changed, thus allowing the iLock Components to refresh their cache when necessary.

4.1 jLock

The jLock Component provides security solutions for applications running on J2EE Application Servers, servlets or standalone Java applications. It provides two major features. The first is the Java iLock Interfaces (JII), and the second is the JAAS Login Module.

For more information on the functionality and operation of the jLock Component, please refer to *iLock Security Services: jLock User's Guide*.

4.1.1 JAAS

The JAAS Login Module is an implementation of Sun's JAAS (Java Authentication and Authorization Service) interface that uses the Security Center as the authentication server. Existing JAAS applications can use the JAAS Login Module without any programming changes.

Since Sun's JAAS authorization component is not scalable to large systems, a JAAS Access Controller component is available which provides scalable authorization technology that can make access decision for secured resources.

The more functional JII described below is also designed to integrate with the JAAS Login Module. In this way, systems can use JAAS-based authentication and the scalable JII-based access control mechanisms.

4.1.2 Java iLock Interfaces

The Java iLock Interface (JII) is a collection of Java classes that provide authentication, administration and access control functionality. The JII can be used by any application written in the Java programming language. It is suitable for standalone applications, servlets, EJBs and even CORBA-based applications written using Java.

The authentication technology supports authentication via User ID and Password or authentication via the operating system. It uses the Security Center as the authentication server.

The administration functionality supports the definition of new secured resource, security policies, users and security attributes. This allows an application system to create security policies, create secured resources, associate security policies with resources, register new users, create new security attributes and associate security attributes with these new users.

The access control functionality provides access decision security functionality equivalent to that provided by the OMG's Resource Access Decision Facility (RAD) specification. The JII makes these access decisions based on resources and security policies that are defined in the Security Center. The JII is also capable of using the RAD Component of the orbLock Component to make these access decisions.

This access decision support is not only simpler to implement but is also more robust and scalable than the primitive access control mechanisms provided by application server vendors. It allows a Java-based application to control access to any resource that can be named. This could include databases, data records and so forth.

In addition to the basic access decision functionality, the JII also supports the development of custom policy evaluators that can be used in making access decisions and the development of a custom service that can manipulate user privilege attributes when access decisions are made.

4.2 webLock

The webLock Component provides security solutions for Web Resources - static Web pages, CGI scripts and Web-based applications written in Java. The webLock Component is compatible with many popular Web servers including Apache HTTP Server (1.x and 2.x) and Microsoft Internet Information Server (IIS). The webLock Component runs inside a Java Servlet Container that processes requests for a Web server.

The webLock Component consists of components used for authentication of Web users and access authorization to Web Resources. Auditing is provided for all authentication and access authorization attempts. These components can be used independently or together. They utilize the users and resource policies set up in the iLock Security Center.

The access authorization is more robust than the mechanism provided by Web servers and Java Servlet Containers and reverse proxy servers alone. It provides controlling access by a user's Roles, Access IDs and Groups. Policies can be set for Web Resources at multiple levels, such as particular machines and/or ports, down to specific applications or even the parameters an application is being invoked with. Using the iLock Security Center access policies for all of an enterprise's Web Resources can be managed in one place. The same access policies in the iLock Security Center can be used for Web and non-Web-based Resources that need protection, simplifying security management.

For more information on the functionality and operation of the webLock security components, please refer to *iLock Security Services: webLock User's Guide*.

4.3 orbLock

The orbLock Component secures resources found in a CORBA-based environment and is composed of three major sub-components:

1. CSIv2 for orb2 for Java
2. CORBA Security Service (CSS)
3. Resource Access Decision (RAD) Facility

4.3.1 CSIv2 for orb2 for Java

The Object Management Group has developed two specifications that can be implemented to provide secure CORBA environments:

1. Common Secure Interoperability (CSIv2). This specification defines standard protocols for implementing security functionality in secure CORBA environments.
2. CORBA Security Service (CORBASec). This specification defines standard interfaces that applications and administrative tools can use to interact with an underlying security service.

The CSIv2 sub-component of orbLock supports the Common Secure Interoperability specification (CSIv2), as implemented by 2AB's orb2 for Java product. This feature allows the authentication of users with either TLS/SSL or an IETF-based User ID and Password standard. In addition, it provides access control for CORBA IDL-defined interfaces and operations.

For more information on the on the CSIv2 sub-component of orbLock, please refer to *iLock Security Services: orbLock User's Guide*.

4.3.2 CORBA Security Service (CSS)

The CORBA Security Service (CSS) sub-component of orbLock provides a CORBA security service that authenticates clients, delegates client credentials and controls access to CORBA IDL-defined interfaces and operations. This component provides security service software that runs in the same process with CORBA-based clients and/or servers and is primarily useful when using CORBA vendor products that have not yet implemented the CSIv2 specification.

CSS can be used with any ORB implementation which properly implements the portable interceptor technology defined in the OMG's CORBA specification.

4.3.2.1 Security-Unaware Applications

The CSS software runs in the process with both CORBA clients and services and requires no coding changes in any client or server software. Applications that are secured without any coding changes are said to be *security-unaware.* This service provides the following functionality for security-unaware applications:

- For CORBA clients, it obtains user identification information (such as User ID and Password) and creates a secure token that is passed to services to verify the authenticity of a client.
- For CORBA services, it validates the secure token passed from clients to authenticate each client.
- It obtains security attributes (Access IDs, Groups and Roles) that are associated with each individual client.
- It uses a client's security attributes, along with security policies, to allow or disallow access to individual operations and/or objects.
- In multi-tier environments, it passes client identity across tiers, allowing the final target service to allow access to individual operations and/or objects using the identity of the originating client. This process is called delegation.
- It stores audit information about significant security events (access to an operation is denied, for example).

4.3.2.2 Security-Aware Applications

Applications that write code to control some aspect of the security environment are said to be *security-aware* applications. The CSS software provides an API with the following capabilities for security-aware applications.

- CORBA services can retrieve the security attributes associated with the invoking client. It may use these attributes to make additional application-specific access decisions (e.g. perhaps with orbLock's RAD Component).
- CORBA services can indicate whether or not an object should be secured, that is, it can force some objects to be secured and others to not be secured.
- CORBA services can specify the name of the object domain that an individual object belongs to.
- CORBA clients can control whether or not its security attributes may be delegated across multiple tiers.
- CORBA clients can control the means by which a user's identity is obtained. For example, it might choose to integrate that functionality in a GUI client program.
- Custom authenticator software can be developed to support a wide variety of underlying security infrastructures. This would include technologies such as SSL, Kerberos or even locally developed security technologies. Customers can do this or 2AB will contract to build custom Authenticator software. This custom authentication software replaces the default authentication software, which supports User ID and Password, which comes with the Security Service.

Note 2AB has already done work developing additional authenticators, such as one that uses X.509 certificates for authentication; therefore, there might already be an available authenticator that can be obtained immediately.

Custom software to look up security attributes associated with a user can be developed to replace the standard mechanism that comes with the Security Service. This might be useful in enterprises that already have a deployed LDAP directory of users and attributes

4.3.3 Resource Access Decision Facility

The RAD sub-component provides applications with a standardized means of controlling access to secured resources. This facility provides applications with an access decision facility for any secured resources that can be assigned a name. As its name implies, this component provides applications a standardized means of determining whether or not a user should be allowed to perform a given operation upon a secured resource. It enables enterprise applications to use common software components to protect application resources and/or functionality.

RAD provides a flexible, rules-based policy engine that implements the OMG's Resource Access Decision (RAD) Facility. RAD provides a mechanism for obtaining access decisions for any named resource. A key aspect of RAD is the ability to administer flexible access policy based on logically defined names. A Resource Name can define a single secured resource or group of resources. Named resources can be anything... Web pages, windows, tabs, menus or buttons, CORBA interfaces and operations, XML documents or confidential data such as health information. RAD provides the ability to implement access control functionality at the level of granularity your enterprise requires.

RAD is designed for integration with existing security infrastructures to enable fine-grain access control. RAD is unique in the fact that it provides a framework for developers to integrate fine-grain access control. The need to secure application-level resources and to integrate access gateways in technology bridges (which is really what an application server is!) has been an ongoing problem for software developers and solutions providers. RAD integrates naturally with any CORBA® security service implementation, but may also be used in environments with SSL, PKI, Kerberos or other authentication mechanisms. RAD can integrate

with any security service that provides authentication and delegation of user security attributes in an n-tier environment. For example, RAD integrates with the CSIv2 and CSS Components described in this guide.

RAD provides a complete implementation of the OMG's Resource Access Decision (RAD) Facility. This OMG specification provides a standardized means of managing protected resources.

4.3.3.1 Basic Functionality

The basic functionality of the RAD Service is to provide access decisions for applications wishing to perform some operation upon a secured resource. Secured resources might be something concrete, such as a database table or row. On the other hand, it might be something that is quite abstract, such as a unique category of information about people and/or customers. Secured resources are assigned unique Resource Names. The internal format of a Resource Name is defined by the OMG's RAD specification. It is a data structure that contains a name referred to as the Resource Naming Authority and a sequence of name-value pairs called a Resource Name Component List.

Note The RAD Component does not permit resource names that have embedded tab or newline characters. This restriction applies to both the "resource naming authority" field and the name_string and value_string pairs in a ResourceNameComponent.

A standard API mechanism is provided that allows applications to request these access decisions. This standard API mechanism is defined by the OMG's Resource Access Decision Facility specification.

Suppose an application needs to secure access to a specific database. It needs to control which users are allowed to **read** and/or **update** the database. To do this, the database must have been assigned a unique name (Resource Name) and that resource name must be associated with a security policy that governs who is allowed to perform the **read** operation and who is allowed to perform the **update** operation. At runtime, the application can then call a method, **access_allowed**, to determine whether or not the access should be allowed. This method takes three parameters. The parameters include the name of the resource being accessed (e.g. the database), the operation to be performed (e.g. update) and the privilege attributes associated with the user making the request. These privilege attributes can be obtained from the underlying security infrastructure (e.g. CSIv2 or CSS). The RAD Component determines the security policies associated with the resource and makes its decision based on the policies associated with that resource, the operation requested and the user's attributes. It returns its decision (true or false) to the application, which is responsible for enforcing the decision (i.e. taking appropriate action to deny the access).

The OMG's Resource Access Decision Facility specifies that a component, called a *Policy Evaluator*, is responsible for making these decisions. The RAD Component comes with a system-default Policy Evaluator called the iLock Policy Evaluator. It uses time-based security policies to make these access decisions. Although this component is completely replaceable, it is anticipated that its features will meet a wide variety of enterprise security needs. Please see the *iLock Security Services: Security Center Guide* for a complete discussion of the security policies used by the iLock Policy Evaluator.

Although it is not necessary, it might be beneficial to understand the basic work being performed by the RAD Component when an access decision is requested. When the RAD Component receives an access decision request, it first checks an in memory cache of resource name policy associations. If the associations are not in its local cache, to communicates with the Security Center Service to obtain it. It then determines if the required security policies reside in its in memory cache. If not, it communicates with the Security Center to obtain them. The internal access

decision engine determines whether or not to allow access. A special notification mechanism allows the RAD Component to update its cache if resources and/or policies are modified. As you might guess, the ability to cache policies and resource-policy associations is critical to providing high performance capabilities.

Beyond the basic functionality of determining access decisions for applications, the RAD Component provides a wide variety of services to help applications manage complex security environments. Applications may also:

- Programmatically define new resources and associate security policies.
- Provide custom software to examine and/or modify users attributes prior to making an access decision.
- Provide custom software to evaluate security policies, effectively giving application the ability to invent new security policy types.
- Provide custom software that determines how decision.

4.3.3.2 RAD Components

The RAD Component is comprised of several sub-components:

- The RAD Service provides the basic access decision functionality expected by applications.
- The RAD Administrative Tool provides security administrators with the ability to manage and/or configure various RAD options.
- The RAD Test Tool provides administrators with the ability to test access decisions. This is important when developing new security policies or investigating problems with the security infrastructure.

In addition to the RAD sub-components, the Security Center and its tools are also essential in managing the RAD environment. Please see the *iLock Security Services: Security Center Guide* for details about the Security Center and its administrative tools.

The following illustrates the services and tools used in a typical environment using the RAD Component.

(Insert Picture from page 50)

4.3.3.3 Deployment Options

Because the RAD and Security Center Services are distributed services, there are a wide variety of deployment options. These options should be evaluated in the context of the deployment environment and the characteristics of the application with which it is being deployed.

The RAD Components can be deployed in one of three major ways:

1. Fully Distributed – using this deployment option, users of the RAD Service communicate with RAD via distributed invocations, and the RAD Service communicates with a Security Center Service via distributed invocations. Using this deployment scenario, many instances of RAD Services can share a single instance of an Security Center Service.
2. Collocated Security Center – using this deployment option, users of the RAD Service communicate with RAD with distributed invocations, and the RAD Service communicates with a collocated Security Center Service via direct method calls. Using this deployment scenario, each RAD Service has it own unique Security Center Service.
3. Client Collocated – using this deployment option, users of the RAD Service communicate with RAD via direct method calls, and this local RAD communicates with a Security Center Service via distributed invocations. Using this scenario, many RAD clients (with a collocated RAD) can share a

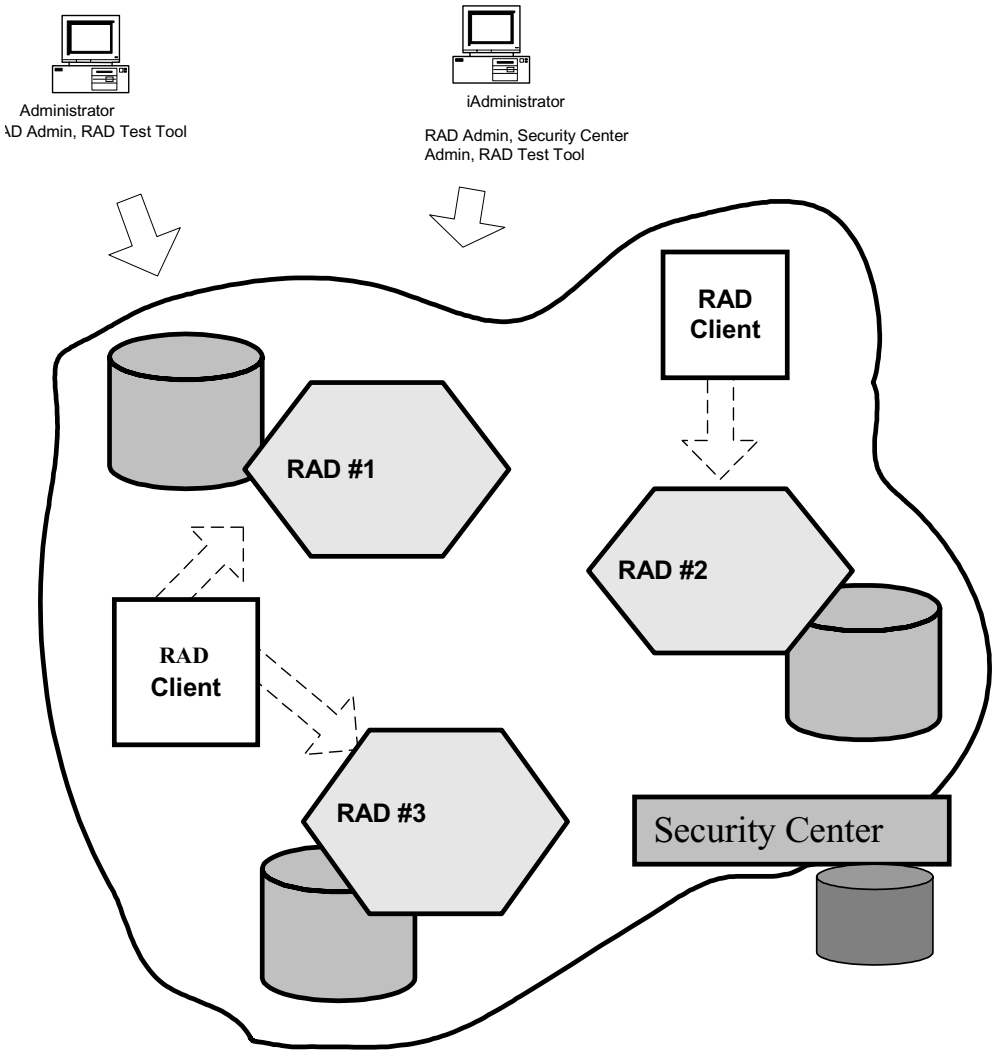
single instance of a Security Center Service. This option limits the functionality of RAD to that of the AccessDecision and PolicyEvaluatorAdmin objects. A more extensive discussion of this option can be found in the chapters describing RAD programing interfaces.

4.3.3.3.1 Fully Distributed Scenario

In the following example of the Fully Distributed option, three instances of the RAD Service are running. There is one client application that is currently using two of the RAD instances. A second client is using only one instance of RAD.

This illustration shows only one instance of the Security Center Service. Each instance of a RAD (via the default PolicyEvaluator) obtains information about security policies and resources from the single Security Center Service. Alternatively, there could have been multiple instances of the Security Center Service. Different RAD instances might then utilize different Security Center instances.

There are two administrators, each capable of administering any of the instances of the RAD Service and the Security Center. The tools the administrators can use include the RAD Administration Tool, the Security Center Administration Tool and the RAD Test Tool.

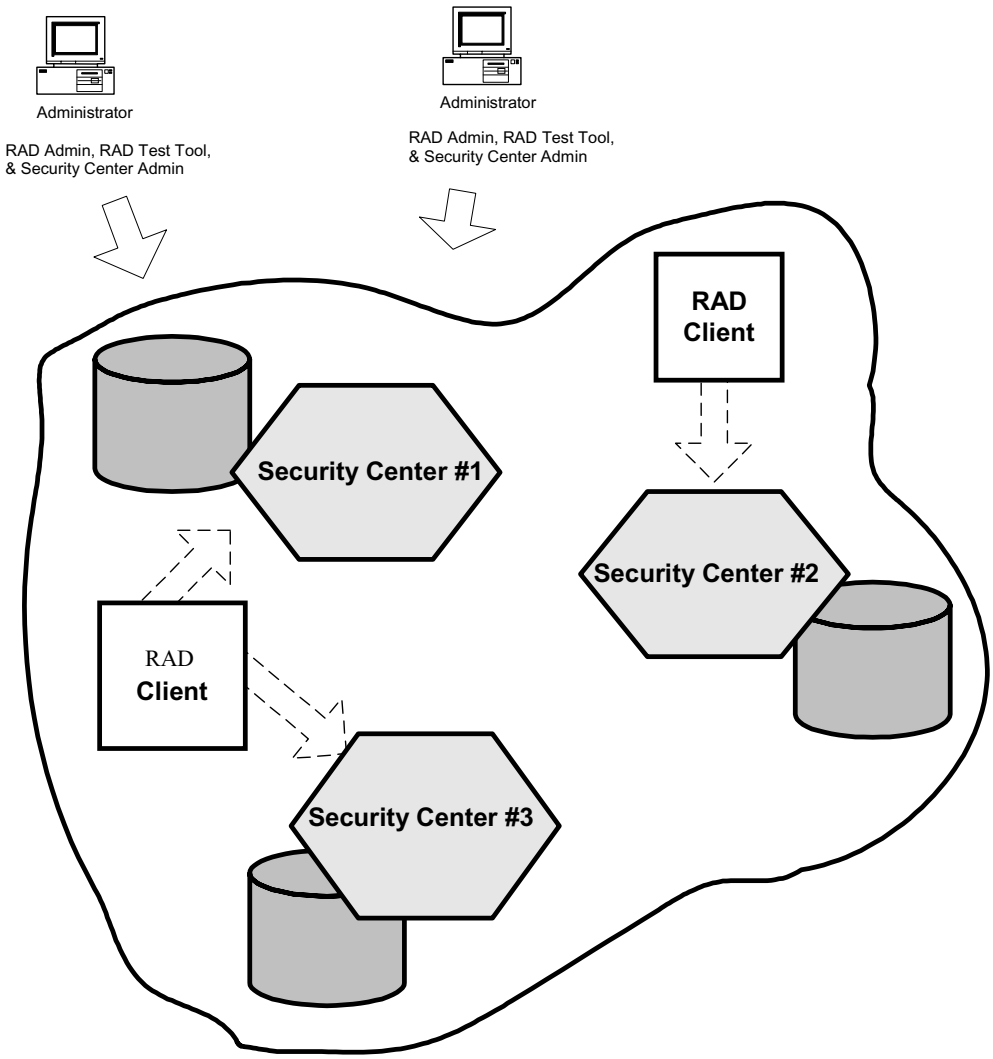


4.3.3.3.2 Collocated Security Center Scenario

The following example illustrates a scenario using the Collocated Security Center option. In this illustration, three instances of the RAD Service are running. There is one client application that is currently using two of the RAD instances. A second client is using only one instance of RAD.

This illustration shows that each instance has collocated a copy of the Security Center and, thus, each instance of RAD has its own unique security center database. Each instance of a RAD (via the default PolicyEvaluator) obtains information about the security policies and resources from the collocated Security Center Service.

There are two administrators, each capable of administering any of the instances of the RAD and/or the collocated Security Center. The tools the administrators can use include the RAD Administration Tool, the Security Center Administration Tool and the RAD Test Tool.



4.3.3.3.3 Client Collocated Scenario

The Client Collocated scenario would look identical to the picture of the Fully Distributed scenario; however, the dotted arrows from the client to the RAD Services would be direct object calls, not distributed invocations.

Chapter 5 Getting Started with iLock Security Services

This chapter provides step-by-step instructions for getting started with the iLock Security Services and assumes that a Windows NT platform will be used; however, other platforms could be easily substituted. You will be introduced to the iLock Components by running some of the demonstration programs that come with the product.

5.1 Install the Product

The iLock Security Services software is packaged on a CD and includes an *Installation Guide*. Follow the instructions found in the guide and ensure that the product has been properly installed.

5.2 Install a License

The Security Center and the iLock Components require a license file. You should have been provided with a license when you purchased the product. If not, contact support@2ab.com. The license file will provide permissions for running the various components that comprise the iLock Security Services. To install the license file, move a copy of the license file that you have been provided to the *lib* subdirectory of the iLock installation directory. This only needs to be done on machines where the Security Center will run.

If the license is not properly installed, or you attempt to use an unlicensed feature, you will be notified when you attempt to use the component.

5.3 Environment Variables

There are three environment variables that must be set on the Windows NT platform. Each of these variables should have been set when running the installation program. If not, use the Windows Control Panel to set the following variables:

Note If you do not wish to make these setting permanent, you may use the Windows NT **set** command to set environment variables in a command-line window.

1. PATH - this environment variable must include the full path name for the *bin* subdirectory of the iLock Security Services installation. An additional requirement for compiling demonstration programs is the PATH variable also includes the *bin* subdirectory of a Java JDK installation.
2. JRE_HOME – this environment variable must be set to the installation path of a Java Runtime Environment (JRE). The JRE specified must be version 1.3 or 1.4.

3. ILOCK_HOME - this environment variable must be set to the path of the iLock Security Service installation directory.

5.4 Run the Demonstration Programs

Each iLock product (jLock, webLock and orbLock) ships with demonstration programs. These are in the *demo* subdirectory of the installation. Please refer to the User Guide for jLock, webLock or orbLock and the *readme.txt* files for detailed instructions on how to run the demonstration programs. There are also whitepapers available in the *docs/Whitepapers* subdirectory of your iLock installation.

Numerics

A

B

C

D

E

G

I

J

L

M

N

O

P

R

S

T

U

V

W