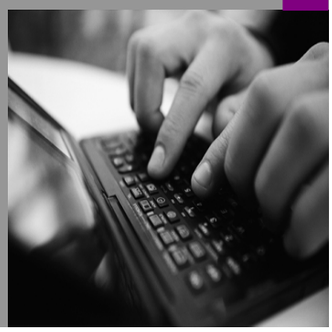

How-to Guide
SAP NetWeaver ‘04

Using System Landscape Directory in Integrative Applications

Version 1.00 – May 2005

Applicable Releases:
SAP NetWeaver ’04

THE BEST-RUN BUSINESSES RUN SAP

© Copyright 2005 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided “as is” without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.
SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.
SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.
SAP NetWeaver “How-to” Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.
Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

1 Scenario

This documentation provides guidelines for organizations which build integrative applications that combine functionality across multiple systems.

The guidelines are intended to help create a robust framework which supports decoupling of physical architecture from business logic, and enables a scaleable approach towards application-to-application communication.

2 Introduction

This documentation describes how to use System Landscape Directory (SLD) to connect custom applications to remote systems.
The guide briefly describes the architecture of the SLD, and explains how to establish connections between applications and various remote systems at runtime.
In addition, it describes how to ensure that your ABAP or Java application can be adapted to the SLD, and explains how to configure the various services and adapters for the SLD.

Both ABAP and Java developers can use the information in this guide to develop applications that need to establish connections to remote systems. Using the information in this documentation, developers will be able to build robust applications which are less likely to be affected by landscape changes.
Moreover, this guide can be used by integration architects as it provides valuable insights about the inter-dependence between the SLD and application development.

Despite the reference to several SLD functionalities and SAP Web Application Server (SAP Web AS) services, the guide does not provide much detail about the technical implementation of the suggested framework. Therefore, organizations that wish to pursue the suggested framework are advised to familiarize themselves with SLD operations and the various SAP Web AS services mentioned in this guide (see section 2.1 for related documentation).

The guide is suitable for any SAP Netweaver ‘04 landscape which utilizes SLD. However, the source code examples are based on SAP Web AS 6.40 SP7 and should be verified for compatibility if other releases are being used.

2.1 Related documentation

2.1.1 System Landscape Directory

Detailed information about System Landscape Directory is available in the SAP Library, for example at:
help.sap.com → *Documentation* → *SAP Netweaver* → *SAP Netweaver 04 (SP9)* → *Application Platform* → *JAVA Technology in SAP Web Application Server* → *Administration manual* → *Server Administration* → *System Landscape Directory*.
Additional guides are available in SAP Service Marketplace at:
service.sap.com/sld.

2.1.2 Configuration Adapter

Information regarding the configuration adapter can be found in SAP Help portal at: **http://help.sap.com** → *Documentation* → *SAP Netweaver* → *SAP Netweaver 04 (SP9)* → *Application Platform* → *JAVA Technology in SAP Web Application Server* → *Administration manual* → *Server Administration* → *Services Overview* → *Configuration Adapter*.

2.2 Overview of System Landscape Directory

Integrative applications are applications which use functionalities that reside in other systems.
One of the inherent requirements of integrative applications is the ability to connect to remote systems. In order to obtain such a connection, the application must either receive it from a "connection provider" or receive sufficient metadata in order to establish a connection on its own from a "metadata provider".
One of the roles of SLD in a complex SAP NetWeaver landscape is to serve as such a "metadata provider".

Example:

An integrative application may reside on the SAP Web AS, connect to an SAP BW system in order to extract data, and according to user interaction update the data in an SAP R/3 system.

2.2.1 SAP NetWeaver Landscape – The role of SLD

A common SAP NetWeaver landscape might include various SAP and non-SAP components. Each of these components may reside on a different server and may even reside on a different network domain.
In addition, any IT environment usually contains several tracks which regulate the release cycle of the applications: a development track, a QA track and a production track are a common combination. Each track contains similar components. For example, each track may contain its own SAP R/3 server, SAP BW server and LDAP server.

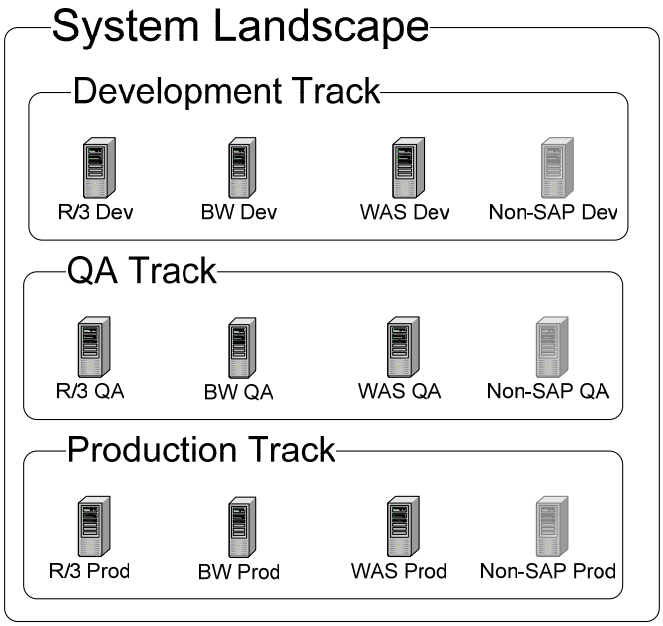

Figure 1 – IT Landscape

Best practice:

One of the roles of the SLD is to manage the transport cycle of software components between the various tracks. For this reason, there should be at least one SLD server which resides outside the tracks, and is aware of all tracks and their components in the IT landscape.

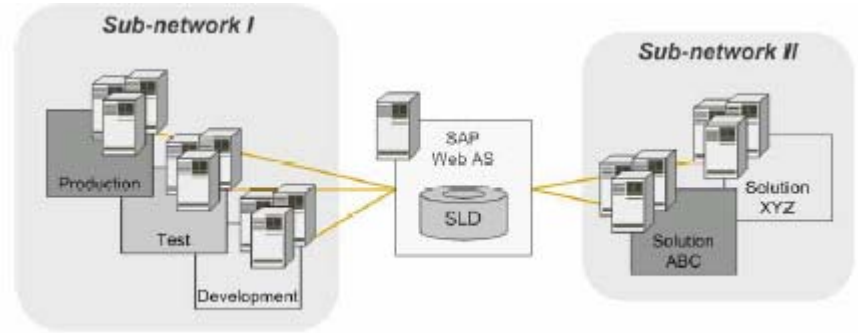

Figure 2 – Central SLD in IT Landscape

However, there could be additional SLDs which are used for a specific track (e.g. Production SLD) or a specific component (SLD for SAP BW). For detailed information about possible SLD architectures, refer to the guide *Post-Installation Guide –System Landscape Directory* available in SAP Service Marketplace at: **service.sap.com/instguides** **nw04** → *Installation*.

3 SLD Client concept

An SLD client is a component which serves as an intermediate layer between applications and SLD. For example, the SLD client queries the SLD server for metadata and uses this metadata to establish connections which are then passed on to calling applications.
When including the connection generation mechanism within the applications, the effort is duplicated. Furthermore, there are numerous advantages to handling resources such as connection through a single point of access (such as pooling, security).

Best practice:

It is recommended to create a single logical service (the service can be implemented as an ABAP RFC or a Java class) which will act as an SLD client and will provide all applications with required connections to remote systems according the metadata extracted from the SLD.
An example of a Java implementation of such a service is provided along with this guide in the file SLDClient.jar which is contained in the zip file containing this document.

3.1 Prerequisites

Configure the SAP Web AS to use a specific SLD as its directory.
The following procedures define how to configure the SAP Web AS for both ABAP and Java environments:

To Configure SAP Web AS Java Stack:

1. Open the SAP Web AS Visual Admin.
2. For each server node, choose "*Services*" → "*SLD Data Supplier*"
3. Under the "CIM Client Generation Settings" tab, fill in the SLD details.

To Configure SAP Web AS ABAP Stack:

Follow the steps described in the topic "Configuring ABAP based clients" in the post-installation guide in SAP Service Marketplace at service.sap.com/instguidesnw04 → *Operations* → *SAP Web AS* → *"SLD User Manual for Web AS 640 / NetWeaver 04"* .

3.2 Data Access Layer

By definition, the SLD client class is logically part of the data access layer of the application. In order to further discuss the structure of the SLD client service, it is necessary to first discuss the responsibilities of the data access layer:
The data access layer is the link between the business logic and the underlying persistence. Therefore, data access objects (or DAOs for short) expose functionality which is business data related (like getMaterialDetail or modifyCustomerName) and performs actions which deal with accessing the persisted data.
The process that the DAOs execute can be divided to the following tasks:

1. Establishing connection to the proxy of the remote system.
2. Performing the correct call to the proxy layer which corresponds to the received request.

While the second task is request-specific, the first task is system-specific since regardless of the data being fetched, accessing the same backend system requires the same connection routine. Moreover, the first task requires the remote system metadata (for establishing the connection), while the second task does not.

Best practice:

Therefore it is recommended to use the following model for the interaction between the Data access layer and the SLD client:

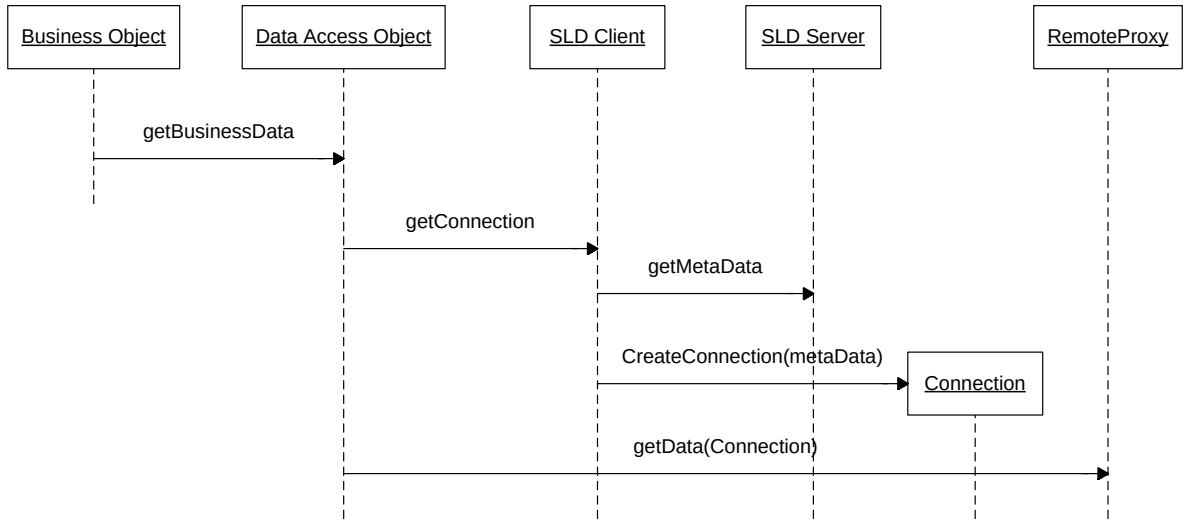

Figure 3 - Sequence Diagram for Data Access Layer

Notice that the diagram is very simplified and is mainly concerned with mainstream data flow between the business object layer and the remote systems. However, since the SLD client is a single point of access for obtaining connections, there are other services it should provide (logging, connection pooling and connection security to name a few) that are not mentioned in the diagram.

Another issue to notice is that the call from the SLD client to the SLD server is done between, possibly, different physical servers. Therefore, the number of times this call is performed should be reduced to a minimum, possibly by utilizing a caching mechanism.

Best practice:

It is recommended to cache system metadata properties in the SLD client in order to reduce the number of remote calls from SAP Web AS to the SLD.

4 SLD Related Development Issues

4.1 Track Identification

An SAP NetWeaver landscape often consists of several tracks, each containing the entire stack of components. When connecting to a remote system it is important to ensure, both for consistency and security reasons, that the remote system resides on the same track as the calling application.
However, since the same code runs on all tracks, the goal of connecting to the correct remote system (in this context “correct” means the one which resides on the same track as the calling application) should be achieved through a configuration mechanism which decouples the logical system from the physical system.
In the “old” ABAP world, such decoupling would have been achieved through defining a logical destination (transaction SM59). By utilizing this method, the program would always connect to the same logical destination (e.g. “MY_REMOTE”) while this logical destination would point to a different physical destination in each track.

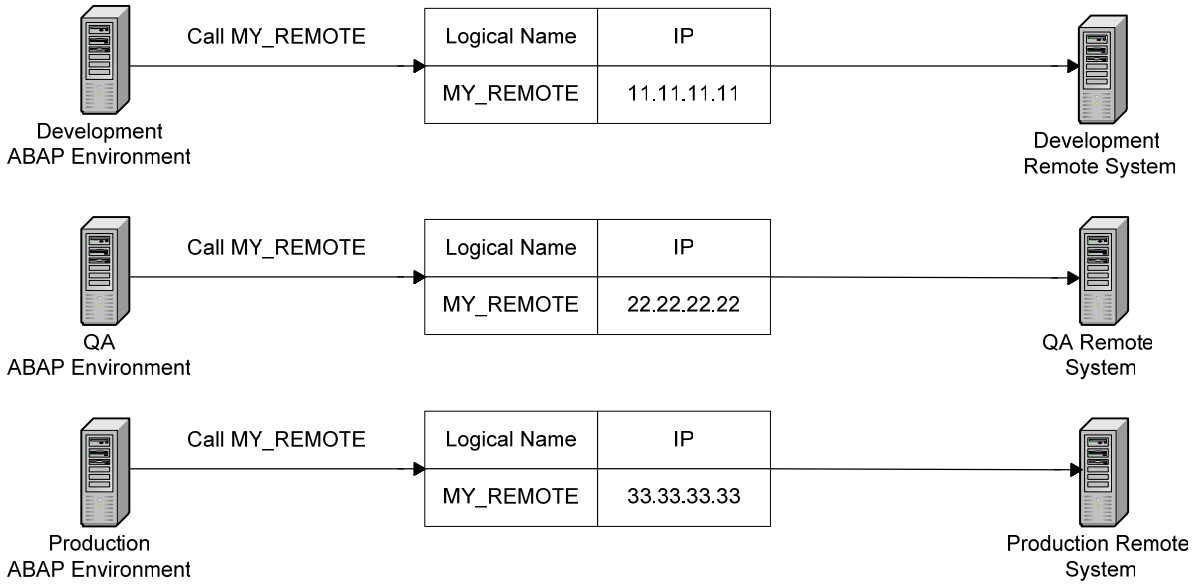

Figure 4 –“Old” Logical\Physical Destination Decoupling in ABAP Environment

The abovementioned pattern was possible since each ABAP system had its own landscape directory (SM59 transaction is a kind of a landscape directory). However, the use of this pattern is not recommended in the “new” SAP NetWeaver landscape since there should only be a single landscape directory.
The solution for this problem is that each deployment should be aware of its track. This track would then be used in order to establish the logical name of the “correct” remote system.

Best practice:

All component systems will be defined in SLD according to the following naming pattern: <SYSTEM_NAME>_<SYSTEM_TRACK>. In addition, every deployment will be configured to be “aware” of its track.
During runtime, calling applications will concatenate the track of the deployment they are part of to the logical system name they require in order to get the full system name as it appears in SLD.

Example:

A development deployment will be configured as development track “DEV”. The SAP BW generic system name will be “MyBW”.
When trying to connect to the SAP BW server in the development environment, the SLD client will lookup the metadata of system “MyBW_DEV” in SLD.

The only issue that is yet to be resolved is how to make a deployment “aware” of its track. Here too the solution depends on whether we deal with a Java or ABAP environment:

Java Environment:

It is possible to use the Configuration Adapter in order to set and retrieve the track for a given deployment (see “Configuration adapter” section in this guide).

ABAP Environment:

It is possible to create a parameter table which will contain a “TRACK” parameter and will be open for changes in all systems (much like the SM59 destinations). This table should be coupled with an RFC which exposes the parameters’ values.

4.2 Connection to a Group of Remote Systems

It is often required to connect to a logical group of systems rather than a single system. For example, such a scenario might occur when the organization employs multiple SAP R/3 systems and would like to gather data from all of them.

Best practice:

A good practice is to enrich the SLD client by adding methods which return collections of connections according to a certain pre-defined logic. This logic should **not** be business logic (since the SLD client is an infrastructure utility), but rather a configuration\landscape logic (such as “get all SAP R/3 systems” or “get all EMEA landscape systems”).

Example:

A common solution for the scenario described above would be to add a getR3Systems() method to the SLD client class.

Even when using the abovementioned methods, it is desirable to minimize the coding changes required to them when the landscape changes (remember! Landscape changes might mean a mere IP or port change so they are not necessarily uncommon). Here too it is possible to make use of the naming convention of the systems in SLD.

Best practice:

Use a common identifier in SLD system names for systems which compose a single logical group. Create methods in the SLD client class which extract the systems according to the identifier. This way, when adding or deleting a system there is no need for a coding change.

Example:

You might use “SAP_BW” as a prefix for all SAP BW systems. For example “SAP_BW_EMEA” for the SAP BW systems which serves EMEA, “SAP_BW_AMC” for the SAP BW system which serves the Americas and so forth.
Then you might create a getBWSystems() method in the SLD Client class which returns connections to all systems whose name begins with “SAP_BW”.

4.3 Calling the ABAP Environment from the Java Environment (and vice versa)

One of the features of the SAP Web AS environment is that the developer has two development environments (ABAP and Java) to choose from on a best-fit judgment. Moreover, the decision does not have to exclusively favor one environment over the other, but rather, allow different components to be developed on different environments while still interacting with each other.

Best practice:

In order to call one runtime environment from the other follow the following steps:

1. Identify the name of the current environment on which you are running.
2. Manipulate the name so it corresponds to the correct SLD that points to the other environment.
3. Get the connection via the SLD client service.

There are several issues that arise from the above flow:

How to identify the current environment on which the code is running?

In ABAP environments, identifying the system on which the code is running can be done by evaluating the SY-SYSID parameter you can find out the system ID on which the code is running.
In Java environments you can use the following code in order to retrieve the cluster name on which the code is running:

```
// obtain the "local" SystemID (SID)
   String systemID = System.getProperty("SAPSYSTEMNAME");
```

Caution

The SAP Web AS supplies a JMX adapter which exposes various management attributes on the server, among which is the cluster name which also uniquely defines the cluster on which the code is running. However, to use the JMX adapter the user in the context must have a J2EE_admin rights. As a result the option to use the JMX adapter should be considered carefully.

How to manipulate the system ID in order to discover the ABAP system ID?
Here too it is recommended to use a naming convention to easily and generically manipulate a Java environment ID into an ABAP environment ID and vice versa.

Best Practice:

For each SAP Web AS cluster installed in your system it is recommended to use the following naming convention for the SLD system name:
<GROUP_NAME>_<CLUSTER ID>_<SYSTEM TYPE>_<TRACK_NAME>.
Where “SYSTEM_TYPE” can be “ABAP_WAS”, “JAVA_WAS”, “BW” or any other component type you might have in your landscape.
By obtaining the CLUSTER_ID from the java.lang.System class, the TRACK_NAME from the configuration adapter and setting the SYSTEM_TYPE to the type which corresponds to the system you wish to connect to, it is possible to lookup the correct system in SLD.

5 Appendix: The Configuration Adapter

5.1 The Configuration Adapter Service

Application properties are application-specific values which can be read during application runtime and can be used as part of the processes in the application. The J2EE specification enables several methods of managing application properties:

- Web.XML for web components
- Environment variables in EJB deployment descriptors for EJBs
- The Java command line –D properties for values which are JVM-specific.

However, when evaluating application properties access mechanism it is important to notice the following characteristics:

- Can the properties be changed in runtime, without a requirement to restart the application or the JVM?
- Does the application properties infrastructure supports server clusters in the sense that the administrator can maintain the values in a single place while affecting the entire cluster?
- Is there a convenient API for application developers, which exposes the application properties?

In order to address all of the abovementioned requirements, SAP Web AS provides a configuration adapter service to its applications. The configuration adapter service is a tool which enables managing, persisting and exposing application properties. It enables modifying the properties at runtime (either through the Visual admin, the Config tool or through shell commands), supports clustered environments and provides a convenient API to the developers.

Best practice:

It is recommended to use the configuration adapter service in order to maintain the track of the application. In order for this recommendation to be effective it is important to be consistent when managing tracks to different applications.

The procedure for doing so is described below:

1. Include a standard Java property file named `sap.application.global.properties` in the META-INF directory of your application EAR, where the *application.xml* and the *application-j2ee-engine.xml* descriptors are also stored.
2. Add to the property file the name-value pair:

```
## Track Property
TRACK=DEV
```

3. Access the track property from your application components using the following model:

```
private static final String PROPERTY_TRACK="TRACK”;

...

//get a handler:
Context ctx = new InitialContext();
ApplicationConfigHandlerFactory cfgHandler =
(ApplicationConfigHandlerFactory)ctx.lookup("ApplicationConfiguration");

java.util.Properties appProps = cfgHandler.getApplicationProperties();

if (appProps==null) {
  //some reaction if no application properties are available.
} else {
  //extract track property
  String track = appProps.getProperty(this.PROPERTY_TRACK);
}
```

By adhering to the above recommendation, an application can connect to the correct remote system (again, “correct” means the instance of the remote application which is in the same track as the connecting application) in the following manner:

1. The application contains a hard coded (possibly through a constant) partial logical name of the remote system (e.g., “R3_System”).
2. The application extracts its own track from the configuration adapter using the above code (e.g., “DEV”).
3. The application concatenates the two strings to construct a full logical name (e.g., “DEV_R3_System).
4. The application passes the constructed logical name to the SLD to obtain the system’s metadata. **Notice:** SLD is **not** aware of the track notion, neither for the calling application nor for the required system.
5. The application creates a connection to the logical system using the metadata.

Naturally, The last 4 steps could be performed by the SLD client for better re-usability.

6 Appendix: Developing and Extending the SLD client

Developers who wish to create their own SLD client should focus on the following packages and APIs:

6.1 Using the Configuration Handler

Use the configuration handler for allowing the application to store and retrieve deployment specific data (such as the deployment's track).

For information regarding programming with the configuration adapter see: **http://help.sap.com/saphelp_nw04/helpdata/en/0d/7fcb4974874767be388007bf9e5c2a/content.htm**.

The configuration adapter API can be found in:
https://media.sdn.sap.com/javadocs/NW04/SP9/j2eeengine/index.html

6.2 Extracting data from the SLD

In order to retrieve the metadata stored in SLD (the basic functionality of the SLD client), the developer has to know ho to connect to SLD and how to browse through the stored data.

Developer's guides for SLD development are only available internally. See:
Http://service.sap.com/landscape → "Media Library – Internal".

The SLD API can be found in:
https://media.sdn.sap.com/javadocs/NW04/SP9/lcrapidoc/index.html.

www.sap.com/netweaver

THE BEST-RUN BUSINESSES RUN SAP