
[Home](#) | [Msg. Solutions](#) | [Dir. Solutions](#) | [Company Info](#) | [Contact Us](#)
[PMDF Support](#) | [IDS Support](#) | [Events](#) | [News](#) | [Jobs](#) | [Directory](#) | [Search](#)

Development and Deployment Of Robust Architecture-Independent Directory Synchronization Across Heterogeneous Directory Environments

**A Technical Overview
From
Innosoft International, Inc.**

Published: May 15, 1998

**By Peter Coates
Senior Software Developer
Principal Architect of Innosoft's PMDF-DIRSYNC Product**

Table of Contents

- ✧ **Why Synchronize Directories?**
 - ✧ **Transparency to the end-user**
 - ✧ **Automatic Operation for efficient administration**
 - ✧ **Robustness and Extensibility for handling diverse directories**
- ✧ **The Challenges of Diverse Multi-master Directory Systems**
 - ✧ **Directory Diversity**
 - ✧ **Multiple Authoritative Masters**
- ✧ **Architectural Issues in DIRSYNC Design**
 - ✧ **Using E-mail as the DIRSYNC Communications Mechanism**
 - ✧ **The DIRSYNC Process**
 - ✧ **Remote and Centralized DIRBOT Functions**
 - ✧ **"Cooking"**
 - ✧ **"Combing"**
 - ✧ **The Centralized Differencing Process**
 - ✧ **"Serving" The Directory Updates**
 - ✧ **Leveraging Standards-based Interchange Formats**
 - ✧ **Use of Expanded LDIF for External Directory Information**
 - ✧ **Special Formatting Enhancements to Optimize Differencing Functions**
- ✧ **Balancing Robustness & Extensibility vs. "Clever shortcuts"**
- ✧ **Keeping Things Simple for Overall System Administration**
- ✧ **The Bottom Line**

Why Synchronize Directories?

In most of today's enterprise computing environments, directory structures have already become the key repositories for managing indexed information. While most people tend to think of a directory as a special database for containing e-mail user information, there are many other examples, such as telephone directories, human resource databases, DNS servers, etc. Also, within a particular application arena, such as an e-mail user directory, the basic structures and the specific content of directories can take many different forms. Despite the diverse nature of existing directories, it is becoming increasingly important for large organizations to be able to share data between different directory environments, such as to create and maintain universal on-line address books for e-mail users throughout a variety of different organizational units.

Because most directory-based information is inherently dynamic, the challenge of continually synchronizing data between interrelated directories becomes a critical issue for successful enterprise wide integration. In order to effectively support mission-critical operational objectives, a directory synchronization system must include the following characteristics.

Transparency to the end-user

A key reason for directory synchronization is to leverage existing user "mind ware" and familiarity with their local operating environments. Rather than trying to impose a new universal data-naming schema on every workgroup, directory synchronization is intended to help transparently "extend their reach".

Automatic Operation for efficient administration

Today's enterprise oriented directories are typically large, mission-critical, dynamic databases that provide the foundation of day-to-day activity for users throughout the organization. Because directory synchronization systems must dynamically keep up with constant change, they must be highly automated and not require unnecessary human intervention or consumption of scarce system admin resources.

Robustness and Extensibility for handling diverse directories

To span the demands of today's diverse enterprise environments, directory synchronization systems must be designed to provide both the external flexibility and internal robustness for dealing with the differing architectures, data structures, and authoritative verification requirements of widely heterogeneous directories.

The escalating trends toward cross-network deployment of Intranet/Extranet applications, network computers, distributed client/server environments and groupware productivity tools are all combining to increase the demand for unified corporate directories. A common theme for all of these network-intensive applications is their heavy reliance on directory based information about users, systems, and organizations. However, unless these distributed directories can be effectively unified, the required information will continue to remain locked in sub-optimal "islands" of localized data. Ultimately, the widespread deployment of robust and extensible Directory Synchronization (DIRSYNC) systems will help provide organizations with the enterprise-wide foundations needed to support migration to this next level of networked interoperability.

The Challenges of Diverse Multi-master Directory Systems

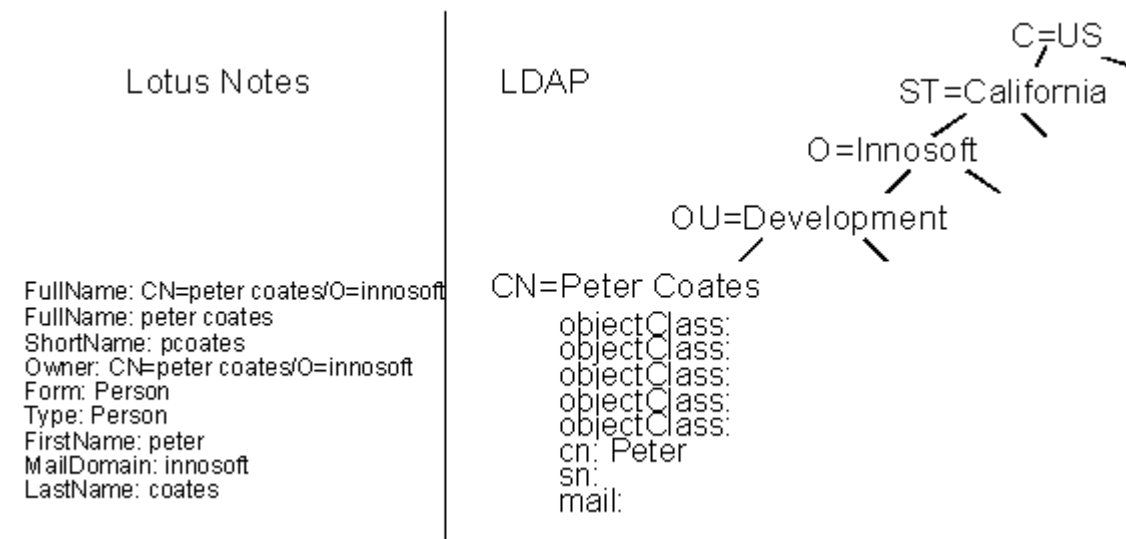
Two of the major problems that have to be addressed to implement effective automated directory synchronization are; 1) the wide diversity of directories and 2) multiple masters that each "own" different parts of the authoritative data.

Directory Diversity

The basic fact of life is that directories differ widely. For instance a few directories, such as X.500 and LDAP, are decidedly tree-shaped, while most others are relatively flat. Some directories, such as Lotus Notes, present a tree-shaped appearance over a basically flat internal structure. More importantly, the internal structures used to represent data can be vastly different from directory to directory.

The key inter-directory data structure issues that directly impact the difficulty of DIRSYNC implementation are 1) the differences in data representation (e.g. field names, attributes and values) and 2) the presence or absence of a unique identifier scheme (e.g. LDAP's Distinguished Name (DN) convention).

Inter-directory differences between attribute names require the DIRSYNC system to conduct some level of data-mapping to create a common representation before it can perform the comparing, reconciling, differencing and updating functions, which are central to data synchronization. For example, LDAP directories use an object-oriented tree structure, with the Distinguished Name represented by the unique path to reach that entry, whereas Lotus Notes uses a structured document to represent each unique entry, with no overlaying tree structure to the database.



The task of inter-directory data mapping can be fairly straight forward, for example mapping two similar LDAP-based e-mail user directories, or it may be quite complex, such as mapping a variety of relevant data attributes from a home-grown human resources directory into a Lotus Notes groupware directory. Special difficulties also arise when trying to synchronize very attribute-rich directories (such as Lotus Notes with dozens of attributes per entry) and relatively simple directories (such as old MS Mail with 3-4 attributes per entry).

Obviously, the establishment of a Distinguished Name or some other unique identifier schema is a

fundamental requirement for implementing an effective DIRSYNC system. While most directories employ at least some sort of unique identifier, some directories actually make no provisions to avoid duplicate records. Since the absence of a distinguished name is a structural impediment to synchronization, the DIRSYNC system will have to create or substitute some sort of primary key (such as an employee number) for handling directories that allow duplicate entries.

Multiple Authoritative Masters

In today's heterogeneous enterprise environments, the legacy development of most directories has been an outgrowth of evolving workgroup level requirements, rather than a top-down integrated design effort. Many such directories contain information that is intrinsically tied to the local functional activities, therefore requiring local authoritative control over the data. Rather than imposing a single authoritative master over all directory information, most enterprises need to find practical ways of optimally supporting synchronization across multiple, distributed authoritative masters.

In such a heterogeneous, multi-master environment, the DIRSYNC system not only has to support and integrate updates from different authoritative masters, it has to recognize and automatically correct inappropriate update attempts from non-authoritative masters. For example, in an environment that combines both MS Exchange and Lotus Notes, the DIRSYNC system needs to propagate changes to a Notes user from within Notes but to recognize as a "mistake" any attempted alteration to a MS Exchange user generated from within Notes. In this situation, the DIRSYNC system would need to make an automatic correction by instead propagating the authoritative information from the MS Exchange master directory.

Architectural Issues in DIRSYNC Design

The following sections will outline some of the critical issues in designing a DIRSYNC architecture that is robust and extensible enough to handle today's heterogeneous, multi-mastered directory environments. In addition, the discussion will provide some relevant examples as to how these issues have been specifically addressed in the development of Innosoft's PMDF-DIRSYNC directory synchronization system.

Using E-mail as the DIRSYNC Communications Mechanism

The choice of a method for transporting directory information requires a balance of both interoperability and reliability. In addition, it has to allow for practical implementation in diverse heterogeneous environments. For instance, building direct-connection data transport mechanisms between all directories might provide the ultimate in reliability but would likely represent an unreasonable investment in both initial development and ongoing maintenance resources. The extensibility of such a mechanism would also be greatly limited because specific links and queuing mechanisms would have to be created for each new directory added or for changes in directory locations.

On the other hand, the use of e-mail to transport directory information provides optimal flexibility, since it doesn't matter if the directories being synchronized reside on the same machine or on different continents. In addition, standards for structured e-mail content, such as MIME, can provide built-in mechanisms for transporting disposition instructions right along with the directory data.

However, e-mail is also subject to being delayed, duplicated or potentially even lost. Since directory

synchronization must constantly handle incremental changes, which are cumulative in nature, the duplication or delay of any single mail update can be potentially damaging. This means that the overall strategy for synchronizing data must be able to handle the non-delivery and/or duplicate delivery of directory updates. In addition, because e-mail is relatively easy to forge, the e-mail transport mechanism used for DIRSYNC must contain a secure signature system to prevent unauthorized changes.

The two most obvious methods for avoiding e-mail duplication and/or omission problems would be to either 1) impose a rigid sequence numbering mechanism on all DIRSYNC agents or 2) maintain complete central databases for determining changes. Innosoft's PMDF-DIRSYNC addresses this issue through a combination these two approaches, by sending complete directories to a central point and returning only the differences. Duplicate updates are also detected and avoided by using a simple "cookie" mechanism that is far less complicated than a rigid sequence numbering scheme.

The DIRSYNC Process

Essentially the PMDF-DIRSYNC architecture consists of a number of Dirsync Agents, one per directory, that extract directory information and communicate it via e-mail to one or more Dirsync robots (DIRBOTS), which conduct all of the data conversion and authoritative directory differencing. After the centralized synchronization is complete the DIRBOT returns only the changes to the Dirsync Agents, which then insert them into the local directories.

To ensure the overall integrity of the synchronization process the system always contains a single central DIRBOT. However to optimize performance and resource utilization, the system generally also uses a number of peripheral DIRBOTS to perform some pre-synchronization functions locally. For instance, typically the most intensive part of the process is the conversion of local directory data into a form that can be compared across all directories. In most instances it makes greater sense, from an overall performance standpoint, to perform this data conversion in a distributed mode on local DIRBOTS prior to transporting the data to the central DIRBOT. In addition, if the directory conversion process includes address mapping, it should always be conducted on the same local machine that does the mapping for e-mail.

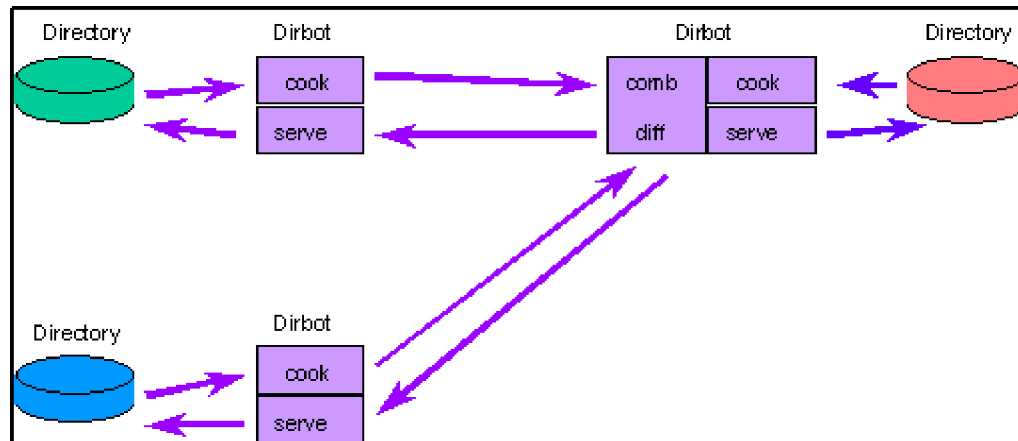
The PMDF-DIRSYNC system always sends complete directories upstream from the Directory Agent to the DIRBOT(s). However, to optimize performance, the DIRBOTS return only the required changes to the Directory Agents. In order to avoid the posting of duplicate updates, the remote Dirsync Agent generates a unique "cookie" and sends it upstream with the directory data. After the centralized differencing and synchronization process, the cookie is returned with the updates. When the updates are applied, the cookie file is deleted. If the cookie does not match the one stored on the remote system or if the cookie has already been deleted, the updates are not applied and an error message is sent to the person in charge of the directory. Because the entire remote directory is sent to the upstream DIRBOT(s) in each cycle, if a set of changes is ever missed or not applied, it will be automatically corrected on the next cycle.

Remote and Centralized DIRBOT Functions

Working together, the remote and central DIRBOTS are essentially the heart of the DIRSYNC system, handling critical functions, such as:

- ✦ Data conversion & mapping ("cooking" the submitted directory)
- ✦ Identification of authoritative entries ("combing" the directory data)
- ✦ Reconciling & differencing (the central synchronization functions)

⚡ Re-conversion and mapping directory updates ("serving" the results)



"Cooking"

The cooking function is performed on directories as soon as they are received by the distributed DIRBOTs. The input from the Dirsync Agent to the DIRBOT consists of the complete raw directory whereas the output is the same data represented in a "cooked" format. "Cooking" essentially consists of converting directory specific LDIF (Lightweight Directory Interchange Format) files to a common form, which can then be compared by the central DIRBOT.

"Combing"

A "side effect" of the cooking process is that the authoritative attribute can be added to some records. This attribute then controls the "combing" process, which takes place immediately after the "cook" process. The "comb" process takes as input the output from the cook process and extracts just those entries marked as authoritative. After this process the DIRBOT holds two versions of the directory: the complete directory and one that contains only the authoritative data.

The Centralized Differencing Process

Once all of the required directories have been received by the central DIRBOT, the differencing (diff) process begins. The first action of the diff process is to concatenate all of the authoritative records to produce an authoritative version of the whole database, as it ought to exist everywhere. The second step is to compare the individual cooked directories against the new master to produce an individual delta LDIF file for each directory. Basically, the delta LDIF file represents the changes that need to be applied to each individual directory to bring it back into sync with the authoritative master.

"Serving" The Directory Updates

The delta LDIF file is then typically sent back to the remote DIRBOT for "serving" into the directory-specific format, prior to sending it to the Dirsync Agent for posting to the directory. Serving is significantly more complicated than cooking because it involves processing of delta LDIF files rather than the more straightforward absolute LDIF files from the original directories.

Leveraging Standards-based Interchange Formats

As indicated above, the obvious starting point for the DIRSYNC data interchange format is LDIF. With the rising tide of LDAP adoption, LDIF has become almost a de facto standard for accessing directory information. However, building in optimal DIRSYNC expandability and performance requires some special extensions and modifications to the standard LDIF format.

Use of Expanded LDIF for External Directory Information

In order to support widely heterogeneous directory synchronization, the LDIF format needs to be slightly expanded to also encompass non-LDAP and/or non-X.500 formats. For instance, LDAP and X.500 directories always start with a DN field. As previously indicated, many other directories do not use DN fields and some do not even use unique entry identifiers. For cross-compatible directory synchronization, PMDF-DIRSYNC has established the first LDIF attribute as the entry's unique identifier, but does not arbitrarily require it to be a DN field. This enables the local directory to use a potentially more appropriate internal unique identifier, such as an employee number. Specialized utilities automatically carry out the conversions from standard LDIF to the nearLDIF expanded format so that internal DIRBOT processing format changes can remain completely transparent to the external directories.

Special Formatting Enhancements to Optimize Differencing Functions

In another key deviation from vanilla LDIF formatting, PMDF-DIRSYNC internally carries out the differencing function using a specialized format for optimal processing performance. Essentially, the accumulation and concatenation of results from the differencing function can be greatly enhanced if the format of the data being compared exactly matches pre-defined templates for the outputs. The inputs for each attribute then can simply be referenced sequentially by Value, with the diff results fed directly into the pre-determined output template.

Essentially, in the first instance, the PMDF-DIRSYNC architecture has opted to "open up" the LDIF format to make it more adaptable to gather in directory data from across heterogeneous directories. In the second instance, PMDF-DIRSYNC has elected to "distill down" the LDIF format in order to streamline and optimize the internal differencing process. However, across the board, the architecture relies upon LDIF's wide acceptance as a standard in order to provide the underlying flexible foundation for extending and optimizing the overall DIRSYNC system.

Balancing Robustness & Extensibility vs. "Clever shortcuts"

In the design of an overall directory synchronization scheme, the system architecture must constantly balance the temptation toward "clever shortcuts" against the overriding imperative to maximize both robustness and extensibility. For instance, if one knew that the only application for the directory synchronization system would be to maintain a group of uniformly structured Lotus Notes directories, one could obviously eliminate some of the cooking and uncooking data conversion processes. Unfortunately, this shortcut would totally preclude extending the DIRSYNC system to encompass a heterogeneous environment beyond the narrowly structured Notes applications.

Or, in another instance, one might even be tempted to minimize the data transmission load by having the remote directory agents transmit upstream only the data that have been changed locally. The danger here is that there is no built-in mechanism for self-correction if the local information begins to drift incrementally out of synchronization. The loss of a single set of directory updates can begin a subtle

chain-reaction of non-synchronization, which may not be caught until its symptoms show up as operational problems for end users. Alternatively, the routine transmission of the entire directory to the DIRBOT ensures a comprehensive reconciliation and differencing process on an ongoing and recurring basis. If any set of updates are missed, the full-directory comparison process will automatically catch and correct the situation on the next update cycle.

Keeping Things Simple for Overall System Administration

Given the myriad of demands already placed upon system administration staff resources, it is imperative that the Directory Synchronization process should not require unnecessary staff time investments, either to run the sync process or, worse yet, to troubleshoot erroneous results. Fortunately, despite the internal complexity of the DIRSYNC process, configuring a Directory Synchronization system can be relatively straightforward because it falls naturally into several manageable pieces.

The first step is to identify which directories are to be synchronized and to decide upon a common directory format for use in the DIRSYNC process. For simplicity's sake, it is useful to consider the underlying requirements for each attribute when developing the synchronization scheme. Not all attributes in every directory have to be synchronized across the enterprise. In many cases, opting not to synchronize "purely local" attributes can significantly reduce the overall task of transporting and comparing data. When trying to synchronize a mix of directories that have widely varying numbers of attributes, one of the primary implementation decisions involves which attributes to include and which to ignore.

The next step involves determining where each of the Dirsync Agents and DIRBOTs should reside. It generally makes sense to have the Dirsync Agents be co-resident with their directories and to place remote DIRBOTs close to the directories they support, however this may not always be the best option. Some parts of the directory synchronization process, such as cooking and serving the data, can be quite CPU intensive, making them good candidates for distributed processing. Regardless of resource issues, it also helps to locate the distributed DIRBOTs close to their assigned directories in order to maintain the appropriate contexts for address mapping. Ultimately, there should be only one DIRBOT assigned to handle all of the central differencing functions and every directory must have a unique name for use by this central DIRBOT.

During actual DIRSYNC operations, the use of e-mail as the communications mechanism provides additional benefits by allowing transmission of disposition instructions along with the transported directory data. For instance, when an LDIF or delta LDIF directory is moved via e-mail, a MIME Content-disposition header contains all of the appropriate commands to initiate action by the receiving DIRBOT. The header information would normally include an Action (e.g. "cook", "diff", "serve" or "apply"), a Name (e.g. the source directory's unique name), a Cookie (the identification token generated by the originating source), and a Key (the signature of the data). For security purposes, the Key is actually a unique hash of the transmitted data, seeded with a shared secret known mutually between the sending and receiving entities. In essence, the DIRSYNC communication system is designed so that every e-mail data transmission is a complete object, containing all of the relevant information needed in order to be recognized, authenticated and acted upon by the receiving entity.

Taken together, 1) the distributed object nature of the DIRSYNC architecture, 2) the use of flexible e-mail communications, 3) the leveraging of X.500, LDAP and LDIF standards, and 4) the reconciling of entire directories rather than just incremental changes, all combine to protect the system administrator

from undue "touch" management of the ongoing DIRSYNC operations. Once configured, the Directory Synchronization system is designed for simple automated operation, with the built-in capacity to inherently correct incremental errors in each subsequent update cycle.

The Bottom Line

Despite the increasing adoption of directory standards and access formats, such as X.500 and LDAP, the enterprise directory environment is likely to continue to remain a largely diverse and heterogeneous arena for the foreseeable future. Part of this has to do with the natural inertia of legacy implementations combined with the organizational investment in users' "mindware" and familiarity with the existing systems. Furthermore, the inherently diverse nature of different applications will continue to give rise to disparate directory implementations. For instance, even if LDAP-based object naming structures become widely adopted for new applications, one would still expect the directory structure for a human resources database to be vastly different from that for a highly-integrated groupware e-mail, scheduling and calendaring application. In addition to these continued differences in directory structures, the need for workgroups and sub-organizational units to maintain local control over their own data will continue to require some degree of distributed multi-master authoritative directories throughout the enterprise.

These ongoing requirements for both directory diversity and multi-mastering will serve to constantly increase the demand for enterprise-oriented directory synchronization systems that can provide both the robustness and extensibility for spanning widely heterogeneous environments. In order to meet these diverse requirements, tomorrow's Directory Synchronization systems will have to combine object-oriented data flow, standards based exchange formats, operational simplicity, and reliable performance. In addition, they will need to be highly configurable to each organization's specific requirements, while leaving ample room for future adaptation as directory structures evolve and change.