

***How Nexaweb Solves Connectivity  
Complexities with our IMB and Desktop Client***

*Technical Brief*

Copyright © 2007 by Nexaweb Technologies, Inc. All rights reserved.

Nexaweb is a registered trademark and Internet Messaging Bus is a trademark of Nexaweb Technologies, Inc. All other marks are property of their respective companies.

No part of this publication may be reproduced in any form or by any means, or stored in a database or retrieval system, without the prior written consent of Nexaweb Technologies, Inc.

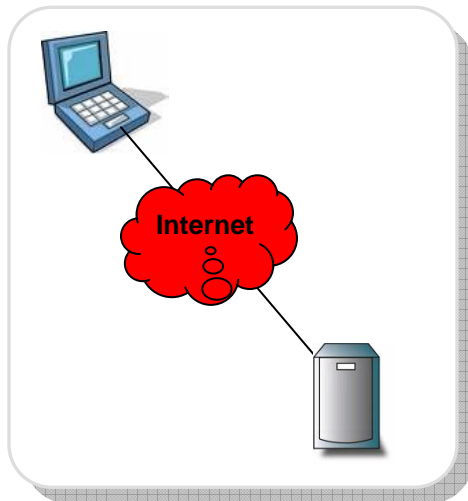
The information contained in this document is subject to change without notice. Nexaweb Technologies, Inc. assumes no responsibility for any errors that may appear.

The software/microcode described in this document is furnished under a license, and may be used or copied only in accordance with the terms of such license. All information, subsystem hardware, and applications described in this document were developed by Nexaweb Technologies, Inc. Nexaweb Technologies, Inc. retains all applicable copyrights therein.

Nexaweb Technologies, Inc. makes no expressed or implied warranties in this document relating to the use or operation of the products described herein. NEXAWEB TECHNOLOGIES, INC. EXPRESSLY DISCLAIMS THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. In no event shall Nexaweb Technologies, Inc. be liable for any indirect, special, inconsequential, or incidental damages arising out of or associated with any aspect of this publication, even if advised of the possibility of such damages.

## Overview

Connectivity issues have long been a focus of application architects as they build their desktop or web-based applications. Understanding of the basic operating parameters of the network is a must to ensure an application remains functional during different modes of connectivity. Applications are never truly connected a 100% of the time; this becomes even more true as applications are deployed over the web. Nexaweb understands this point and we have built into our platform the ability to handle different modes of connectivity as a core feature.



**Ever wonder why the Internet is represented as a cloud?**

## Occasionally Disconnected

In this mode the client must be connected to the server to start and relies on server resources to begin, including user interface, business logic and data for initial functionality. This is the traditional Internet model. Once connected, the client may become disconnected from the server; however, it will remain functional based on its state at connection loss. Simple HTTP request/response doesn't provide the needed infrastructure to handle the complexities of occasionally disconnected applications. Users of web-based applications have gotten used to just hitting the refresh button to perform the action again, this is ok for simple applications that are just post form data or look at a table of data. Expand that functionality to complex non-linear workflows and large amounts of real-time data, and hitting a refresh button is no longer a solution.

## The Internet Messaging Bus Solution

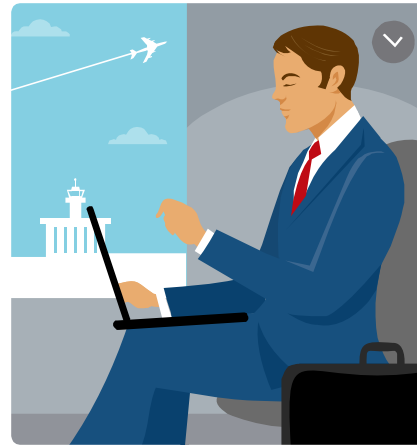
Using Nexaweb's Internet Messaging Bus (IMB) as the communication layer for application development and gives developers a reliable messaging infrastructure that can handle the common network interrupts of the Internet. Nexaweb provides the capability to persist client-side business logic, data, and messages in a disconnected state through in-memory caching of resources. Even when disconnected, the user can still interact with the application as normal.

Applications that take advantage of the IMB for real-time communication to/from the server will notice data pauses when disconnected, but the IMB will cache data until the connection is reestablished. Developers can supply policies to define how the server handles the sending of queued data to the client. This prevents the user from receiving data that may have become stale while they were disconnected.

Actions meant for the server will be queued on the client as well. Once the client reconnects to the server any message that needs to be sent will be transmitted as will any state information that was changed in the disconnected state.

## Occasionally Connected

Connectivity requirements vary greatly across applications. For some applications working online all day is a perfectly acceptable usage pattern. Users that need to be able to work offline for extended periods of time such as a field service or mobile sales force, require applications that can function in an occasionally connected mode. These applications must be able to start either connected to or disconnected from the network. A client that is disconnected at startup requires more application logic to be stored locally including the user interface, business logic and data. The browser will no longer function as the application container. They require a connection to start and provide limited local storage of resources other than cached html pages and images.



**Operating disconnected  
requires greater amounts of  
local resources**

## The Desktop Client Solution

Nexaweb's desktop client provides the ability to deploy our Java rendering engine locally on the desktop. This gives users the ability to start their application offline by launching the desktop application. The Desktop client will utilize resources, data, and business logic that it has been stored on the users system. For developers the desktop client provides two added features on top of those already provided as part of online application:

- **Presence detection** gives the developers the ability to detect when the application comes "online" or goes "offline". Developers can utilize this capability to perform data synchronization and partition where their business logic will be executed.
- **Resource synchronization** removes the problem of keeping your application up to date. This is a common problem for desktop applications because after deployment new versions or patches need to be deployed to users. Standard desktop deployment makes IT organizations reinstall the application, which becomes costly for applications with many end-users. Nexaweb's Desktop client removes the need to reinstall the application by adding infrastructure to keep locally installed resources update date. By update resources in the web application, the Desktop client will detect the change and notify the users that a new version is available, if the user wants, he/she can then restart the application to update the new version.
- **Desktop Integration** is possible with the desktop client, because it is not bound by the same permission constraints of web-based applications, the Desktop client can access the local file system, printers, or any other resources that a normal desktop client has access to.
- **Performance** of your application can also be increased using the Nexaweb Desktop client. Memory allocated to your application can be adjusted to allow for more local data caching. Persistence of local data is typically faster than retrieving it from across the Internet.

Providing developers; presence detection and resource synchronization on top of the functionality already provided by the IMB, the Nexaweb Platform is a solution that can be utilized to build applications that can function either occasionally disconnected or occasionally connected. A large insurance company in Japan is already deploying large scale application that service 38,000 sales representatives, 4000 of which are mobile in on a single application code base.

## Design Considerations

Nexaweb helps application developers and architects overcome many of the challenges of building applications that are required to work in different connectivity states - but there are still some considerations that architects will need to take into account.

- **How will the application be deployed the first time when using Nexaweb's Desktop Client?** Even though Nexaweb's Desktop client handles the synchronization of resources once deployed, an initial install will need to take place. Any of the following ways can be used: zip file, install process, or Java Webstart.
- **What functionality will be accessible while offline?** Depending on your application you may not make the complete functionality accessible offline. The more functionality that is accessible offline means the more control and persistence will be accessible offline.
- **How will Data synchronization be handled?** When the client comes online after a long period of being disconnected, synchronization of data will need to take place and conflicts will no doubt occur. You should therefore build conflict resolution into your application from the beginning to make it easier to handle data synchronization.
- **How will local data persistence be handled?** This is can be done many different ways depending on the complexity of your data model - flat files and databases may be best.
- **What security requirements will the application have?** Because the data and application resources are stored locally, you will also need to think about the security around these resources, locally persisted data, as well as access to the application.

## Conclusion

Whether you have thought about the complexities of how to handle large applications with complex workflows deployed across the Web or not, Nexaweb has, so you can be assured that different connectivity scenarios are covered when deploying application built with the Nexaweb Platform.