

Cirquet™ Solution Suite[⊖]



Technical Whitepaper

Version 1.5 – October 2001

Confidential

© 2001 Improv Technologies, Inc. All Rights Reserved

Introduction

What is Cirquet?

The **Cirquet Technology** is a revolutionary component framework that enables the rapid and cost effective design, development, deployment, management and maintenance of large-scale dynamic network-based distributed applications across a wide variety of platforms, devices and resource capabilities. This is accomplished through the creation of **Cirquets** - lightweight, self-contained, portable *Web services* that may be managed remotely, modified and updated at run-time. Cirquets may be hosted on dedicated peers or servers on the network, dynamically migrated across multiple peers to provide efficient load management, or downloaded to client devices, as needed to provide rich user interactions and distributed processing.

The Problem

Existing products in the distributed component space fall into two major categories: **centralized application servers** and **distributed object frameworks**. Application servers employ a “thin-client” model in which business logic and presentation components are hosted on a centralized server and accessed through a lightweight client (usually a web browser). Distributed object frameworks, like CORBA and DCOM, are used in “fat-client” applications where software installed on each client machine connects to a specific service or set of services hosted on one or more remote devices. In today’s application servers and distributed object frameworks, functionality *must* be designed, deployed and maintained to specifically address the hardware, transport and communications requirements of the underlying network architecture.

The Innovation

Improv Technologies has taken a significantly different approach to the problem through the Cirquet Technology. Cirquet creates a layer of abstraction between component functionality and component communication across the network. This enables Cirquet to employ fast “in-process” communication for components residing on the same peer and cross-platform XML messaging where components are hosted on different peers on the network, without requiring any additional work on the part of the developer. Cirquet’s *dynamic deployment* capabilities enable components to be *distributed, managed and upgraded* over a large number of servers, clients and portable devices from a centralized location, eliminating the maintenance issues associated with today’s distributed application frameworks. Additionally, and equally important, Cirquet supports existing component methodologies, enabling pre-existing components and components created using well known programming techniques to be immediately available as Cirquet-based Web services, eliminating most of the learning curve normally associated with the creation of network-based applications. In addition, Cirquets contain built-in support that enables them to

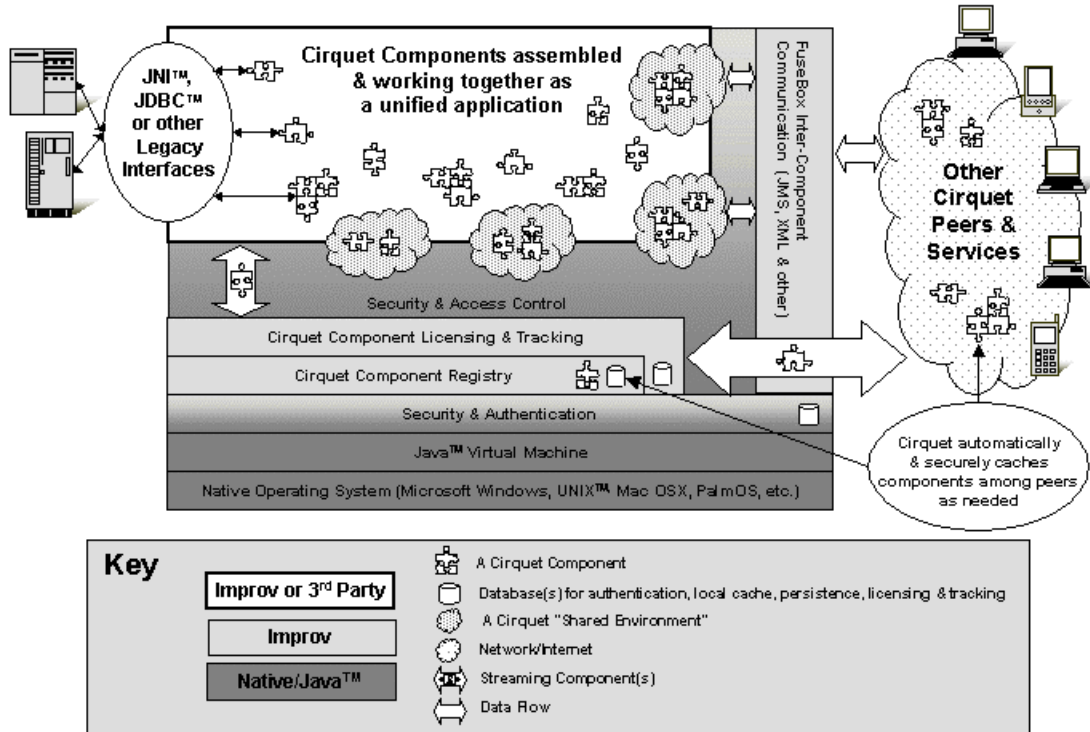
interoperate with standard (non-Cirquet) Web services, such as those proposed by Microsoft's .NET initiative and the Sun ONE Platform.

Empowering Web Services

The vast majority of Web services platforms, tools and technologies, are geared toward the creation and management of a service or set of services running on a single host. While there are tools to enable the creation of applications that make use of a diverse set of independently hosted services, (i.e. Microsoft's Studio.NET) none of these address the issue of managing collections of services running on a wide variety of devices and the interconnections between them. This is where Cirquet comes in. The central tenet of Cirquet is that every distributed application could be composed of a vast array of independent components, running anywhere from a small handful of nodes on the network to thousands or even millions of devices. As such, the ability to centrally manage and maintain the network of services that comprise an application is essential. To date, Cirquet is the only solution that makes this possible.

Cirquet Solution Architecture

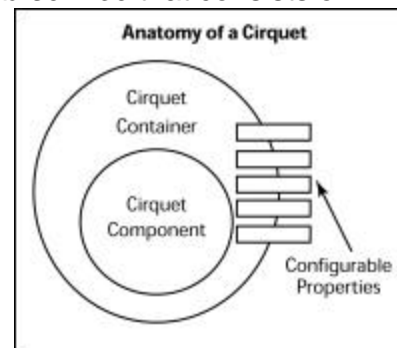
Anatomy of a Cirquet "Peer"



The Cirquet Solution architecture is built upon three elements: **Cirquets**, the **Cirquet Runtime Environment (CRE)** and the **Cirquet Registry**:

A **Cirquet** is a lightweight, self-contained, portable *Web service* that consists of:

- **Cirquet Container** holds the configurable properties of the Cirquet, the code base and class files of the Cirquet as well as access control information. All of these can be dynamically configured at run-time *without* shutting down and recompiling the Cirquet.
- **Cirquet Component** represents the business logic, presentation layer or some other component functionality. This generally comes in the form of a .jar file or Java class referencing any Java language component, including Javabeans™ and EJBs™.



Benefits of Cirquets:

- ***Cirquets can be assembled into distributed applications “on-the-fly”*** – developers create distributed applications by assembling pre-compiled Cirquets in a run-time environment. There is no need to shut down, recompile, and re-install whole applications each time new functionality is added.
- ***Cirquets are easy to re-use*** - because they are self-contained with few external dependencies, Cirquets can be linked to each other in a variety of combinations without the need to write and compile “glue-code.”
- ***Pre-existing components can be easily Cirquet-enabled*** – any Javabean™ or EJB™ can be published to the Cirquet Network.

The **Cirquet Runtime Environment (CRE)** manages the *instantiation* of Cirquets on a local peer, the *communication* between Cirquets, and the *monitoring and tracking of Cirquet usage*. The CRE sits on top of the Java Virtual Machine (JVM) on each of the peers on a Cirquet-enabled network

- **Communication between Cirquets** - Cirquets residing on the same peer utilize fast “in-process” communication. When residing on different peers, Cirquets will communicate using cross-platform SOAP-based messages. The Cirquet framework supports:
 - Synchronous and Asynchronous communications between Cirquets
 - The following networking transports:
 - HTTP – ubiquitous protocol
 - JMS – Java Messaging Service
 - JXTA – lightweight P2P protocol with built in discovery, grouping and security features.
- **Instantiating Cirquets** – The configuration information of a Cirquet and the CRE will determine whether or not a Cirquet is:
 - Persistently cached whereby the Cirquet remains locally stored until the end user or application administrator removes it.
 - Transiently cached whereby the Cirquet would remain locally stored for a set period of time or once the peer is shut down. This would generally be done for security reasons and/or to free up device resources when an application is not being used.

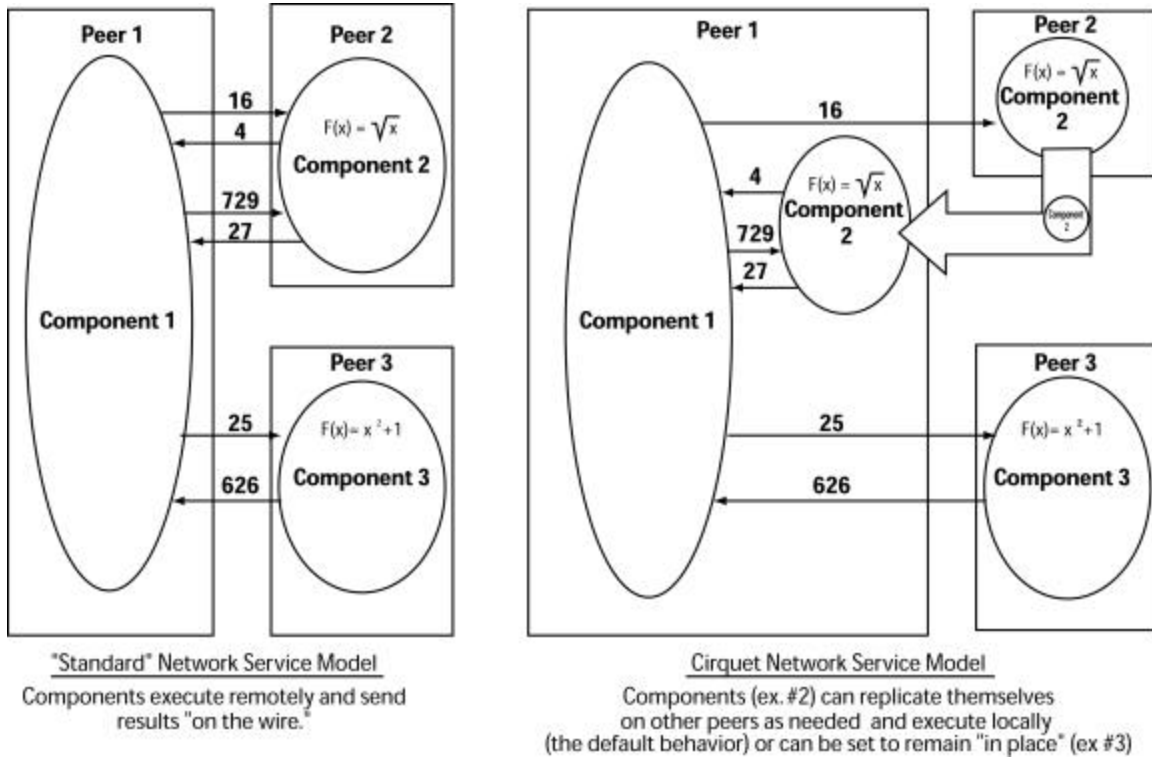
- **Monitoring and Tracking of Cirquet Usage** – all usage statistics across a Cirquet network can be monitored and tracked. There are three ways in which can be done:
 - The CRE can track when and how often a Cirquet is instantiated on a local peer.
 - The CRE can track when and how often messages are sent between Cirquets.
 - The Cirquet can be configured to broadcast an event when it is accessed and used.
- **Benefits of monitoring and tracking:**
 - Developers can use this data to concentrate future development efforts on features needed and requested by users.
 - 3rd party Cirquet developers can use the monitoring and tracking features for billing and licensing purposes.

The **Cirquet Registry** provides a secure, reliable storage and retrieval facility for Cirquets. It provides the ability to perform the following functions:

- **Store (“Publish”) new or modified Cirquets to the network** - The registry enables a developer to add a descriptor to a component that will automatically assign a unique identifier, effectively converting the Java code into a Cirquet Component. Any Java language component, including Javabeans™ or EJBs™, can be published as a Cirquet to the network.
- **Locate and Retrieve Cirquets** – Cirquets are located by querying the registry the local peer for those that match the desired criteria. If none are found on the local peer, then a search is performed on external registries. Once Cirquets are found, the registry returns a list of optimal matches. Cirquets can then be streamed down to the local peer as needed.
- **Cache Components** – components are locally stored the first time they are accessed so that future uses do not require network resources as they are used. An application executing within the Cirquet Framework streams Cirquets from other peers on the network as needed. Once all the Cirquets are downloaded, the peer can disconnect from the network without adversely affecting the performance of the application. Overall bandwidth consumption from application streaming will be smaller and more balanced because of lazy loading and local execution.

- Automatic Cirquet Updating** – Cirquets are cached on multiple machines across the network. When a change is made to a Cirquet, all deployed instances of that component are notified and the changes are immediately propagated to the affected peers.

The following illustration shows how Cirquets can be streamed down to reside on local peers and take advantage of fast “in-process” communications.



Architect Once

It is becoming increasingly more common these days for enterprises to have very mobile and geographically distributed workforces. In order for this workforce to operate effectively and efficiently, they need to share the same applications and data. This puts an increasing demand on the IT organizations within these enterprises to provide the workforce with the same application functionality wherever they may be and on whatever device they may be using (PCs, Laptops, PDA, wireless phones, etc.). Generally, this means that applications have to be designed a number of times to meet the CPU requirements and resource constraints of the various devices employed by the workforce as well as the underlying network architecture. Using Cirquet, developers need only design their applications once.

Device and Network Neutral

Cirquets and Cirquet Applications can be designed and built independently of the device, transport and communications requirements of the underlying network architecture. Any network appliance that is compatible with a Java 2 Virtual Machine (JVM) can be used as a peer.

Deploy on Demand

Deploying applications built with Cirquets is simply a matter of deciding which Cirquets reside on which peer in the network. Developers have a series of options when deploying the **same** application:

- Cirquets can be streamed down on an “as needed” basis. Instead of all the Cirquets that make up a particular application downloading all at once, they may be downloaded when the particular Cirquet functionality is requested. Over time, the whole application may be cached on the local peer taking advantage of fast “in-process” communications.
- Cirquets can be deployed to remain on specific peers across the network for security reasons or to take advantage of the particular resources of the peer. For example, Cirquets that contain sensitive data or functionality (proprietary algorithms, etc) may be deployed only to secure servers and not streamed to local peers. Compute intensive Cirquets may be deployed to remain on peers with adequate processing power.
- For resource-constrained devices such as PDAs and wireless telephones, Cirquets may be streamed down one at a time when the particular functionality is required enabling rich, interactive applications. When it has completed its function or is no longer needed, the Cirquet can be dropped from cache to free up memory without disrupting the application.

Scale at Will

Traditionally, meeting the increased demand on network based applications meant adding additional resources to the network (i.e. more servers). However, with Cirquet, the same resources can be used more efficiently. Because Cirquets can be dynamically configured to reside on any peer on the network, they can be either be:

- **Migrated to underutilized resources** – in order to free up resources placed on a peer due to increased demand for a particular Cirquet or set of Cirquets, developers may choose to migrate less important Cirquets to another peer.
- **Duplicated across numerous peers to provide adequate load balancing** – multiple instantiations of the same Cirquet may reside throughout the network. This lessens the load on any one peer.

Built using Industry Standard Technologies and Practices

Cirquet is built using widely accepted technologies, standards, and software components, such as Java™, JXTA, and XML.

- **Java™ dynamic object linking** – With the Java 2 platform, Cirquet incorporates components at runtime into existing applications, without recompiling the original source code. Cirquet provides a set of core services, as well, with an intuitive interface for the more complex aspects of Java's dynamic programming facilities.
- **XML (extensible Markup Language)** – XML forms the heart of Cirquet's communication model. Component and application meta-data is stored and distributed as XML, and transformed into software components on the client machine. By decoupling a component's properties from its implementation, creators provide multiple component implementations for use on a variety of platforms. This enables application developers to assemble applications tailored to the runtime capabilities status of a particular peer.
- **SOAP (Simple Object Access Protocol)** – the Cirquet Technology uses SOAP as the messaging protocol between all Cirquets. This enables Cirquets to not only communicate with each other, but with any SOAP-based Web services.
- **Standard Transaction Model** – Cirquet provides a distributed transaction model, based on standard ACID (Atomicity, Consistency, Isolation, and Durability) semantics. Therefore, every transaction is guaranteed to complete or not occur at all. Upon completion, the transaction is guaranteed to leave the system in a stable and predictable state. In Cirquet, if a component transaction cannot be completed, due to network failure, Cirquet attempts to access the component through another source. If an additional source cannot be located, the transaction won't initiate and the user is notified of service unavailability. The inherent redundancy of components in Cirquet ensures the highest likelihood of component availability.

Security

Security, authentication, and access control are integral features of the Cirquet Technology. As with any networked environment, the integrity and confidentiality of data are of utmost concern to the user. Cirquet Technology is designed to ensure any data or component transferred between peers must pass through a security layer¹.

- **Java Sandbox** – The Java 2 security model provides fine-grained control over an application's system resource access. Gaining additional access to resources (the local file system, user information, etc.) requires explicit permission from the user.
- **Communication over SSL** – Communication between objects in Cirquet may occur over Secure Socket Layer (SSL), using the Java 2 Secure Socket Extension (JSSE) where necessary. This protocol employs public-private key encryption, generated once an object is published to the network. Any change to a component's data between the time it's published and accessed will result in a failure to decrypt the component at the receiving end.
- **Communication across corporate firewalls** – Communication across corporate firewalls is achieved through a variety of techniques, including transport layer bridges (or port-forwarding), and application proxies, which enable the firewall to direct incoming communication to specific machines on the local network.
- **Encryption** – Components are stored in an encrypted local data store, using the same techniques employed for encrypted communication, which virtually eliminates a user's ability to tamper with cached components.

¹ Because all components may not require the same level of security, access controls and other attributes are highly flexible and customizable. The security layer may never be completely bypassed.

Integrating Cirquet with Existing Systems

Cirquet can be integrated into existing systems, through the use of tools like JDBC™, JNI™, Javabeans™, and Enterprise Javabeans™ (EJB). The sections to follow explain the roles of each and how they help to achieve this.

- **Java Database Connectivity (JDBC™)** connections to commercial databases – Cirquet Applications are easily integrated with commercial databases, using the Java platform JDBC™ API. The majority of commercial database manufacturers, including Oracle and Sybase, currently support JDBC.
- **Java Native Interface (JNI™) integration with non-Java applications and components** – JNI™ provides developers with the ability to integrate platform-specific components and legacy applications written in C, C++, and other programming languages with Java applications.
- **Integration of Javabeans™ and Enterprise Javabeans™ (EJB)** – Cirquet is built on the Javabeans™ component model, making it easy to integrate any Javabeans™ created component into Cirquet. A standard container for EJBs is provided with Cirquet, enabling existing bean deployment into the Cirquet network.

Comparison to Other Component Frameworks

The following sections describe the differences between the Cirquet platform and other existing component based frameworks.

- **Common Object Request Broker Architecture (CORBA)** – CORBA is a standards-based distributed component model proposed by the Object Management Group (OMG). Cirquet does not use a remote server model, like CORBA, so components are executed locally by default. Existing CORBA solutions can be integrated into the Cirquet Platform, through the use of RMI-IIOP.
- **Jini™** – Jini™ network technology provides simple mechanisms for enabling devices to plug together, forming impromptu “communities”. Each device provides services for use by other devices in the community. Like Jini Services, Cirquets are “looked-up” and “discovered” on the network in a fashion similar to Jini, however Cirquet also provides additional services, supporting distributed data storage, dynamic service composition, and collaboration.
- **Microsoft’s Distributed Component Object Model (DCOM)** – Microsoft’s DCOM enables developers to build applications, where various components are executed on different network machines. Unfortunately, COM and DCOM are geared towards those devices using Microsoft operating systems. Furthermore, the use of COM requires a strict prior knowledge of exact components through their unique identifiers (UUID). Aside from the obvious cross-platform and cross-device differences, Cirquet provides a more dynamic runtime coupling between components in a peer-to-peer networked environment.
- **Enterprise JavaBeans™** – Enterprise JavaBeans™ (EJB) support the standard, now ubiquitous, three-tier architecture. It uses a central server, connected to a database, and abstracts many concurrent and database programming and transaction-based computing issues used to construct three-tier applications. Cirquet, on the other hand, is a platform designed to support the development and deployment of custom component-based distributed applications, at runtime. Cirquet seeks to reduce the need for a central server and attempts harness workstation power to create a peer-based environment for enterprise computing. By using Cirquet, enterprise developers can create distributed enterprise solutions, which scale along with company growth.