



Porting a Java application to the HP NonStop Server A practical experience

Peter Lloyd
Webpay Product Manager
eFunds International Limited
Peter_lloyd@efunds.com



The project



Port the Webpay Transaction Server (WTS) and associated application modules to the HP NonStop server platform

- Maintain all existing functionality
- Resulting in a single application source for all supported platforms (ease of maintenance and distribution)
- Take advantage of Himalaya specific capabilities where there is an identifiable value add
- Maintain or enhance scalability, performance, and failure recovery
- Ensure that we keep the architectural integrity of the application



What is the WTS?

- A secure messaging application framework
- A common platform for Internet and wireless based transactional systems
- Secure, authenticated eCommerce payment & messaging solutions for financial institutions
- Easily customizable to cater for specific application requirements
- The foundation for Webpay banking industry specific products

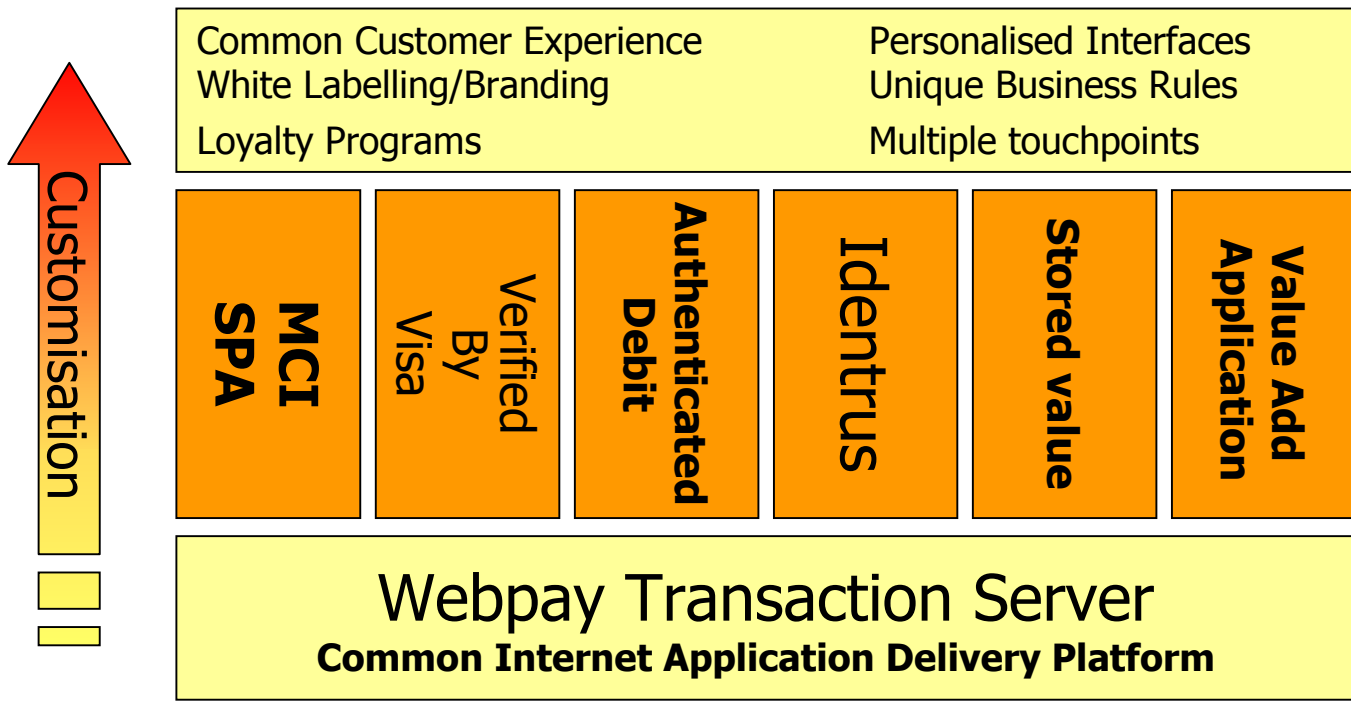


Webpay banking industry specific product set

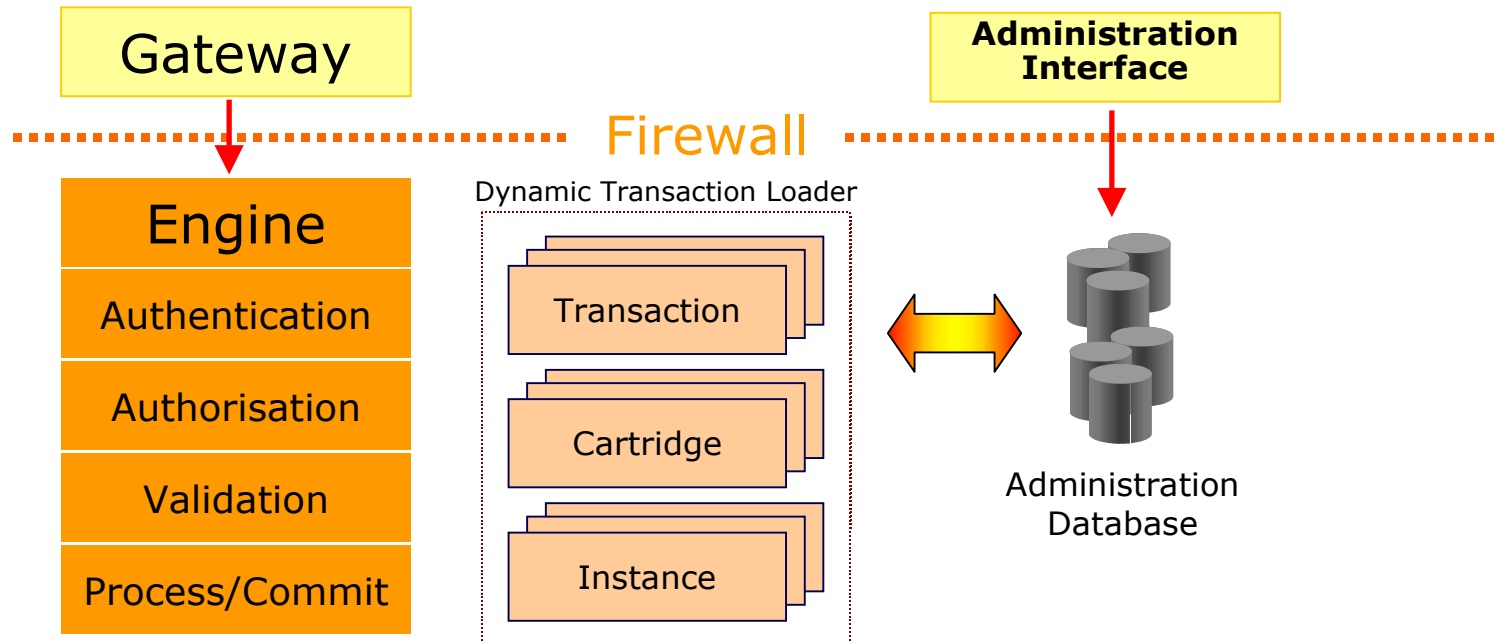
- Payments gateway
 - Credit card
 - Debit
 - Micro payments
 - Loyalty programs
- Authentication Solutions
 - Verified by Visa
 - MasterCard SPA
- Identity and Trust
 - Identrus Transaction Coordinator
 - Eleanor payments initiation
- Secure forms lodgement



WTS architecture



WTS architecture



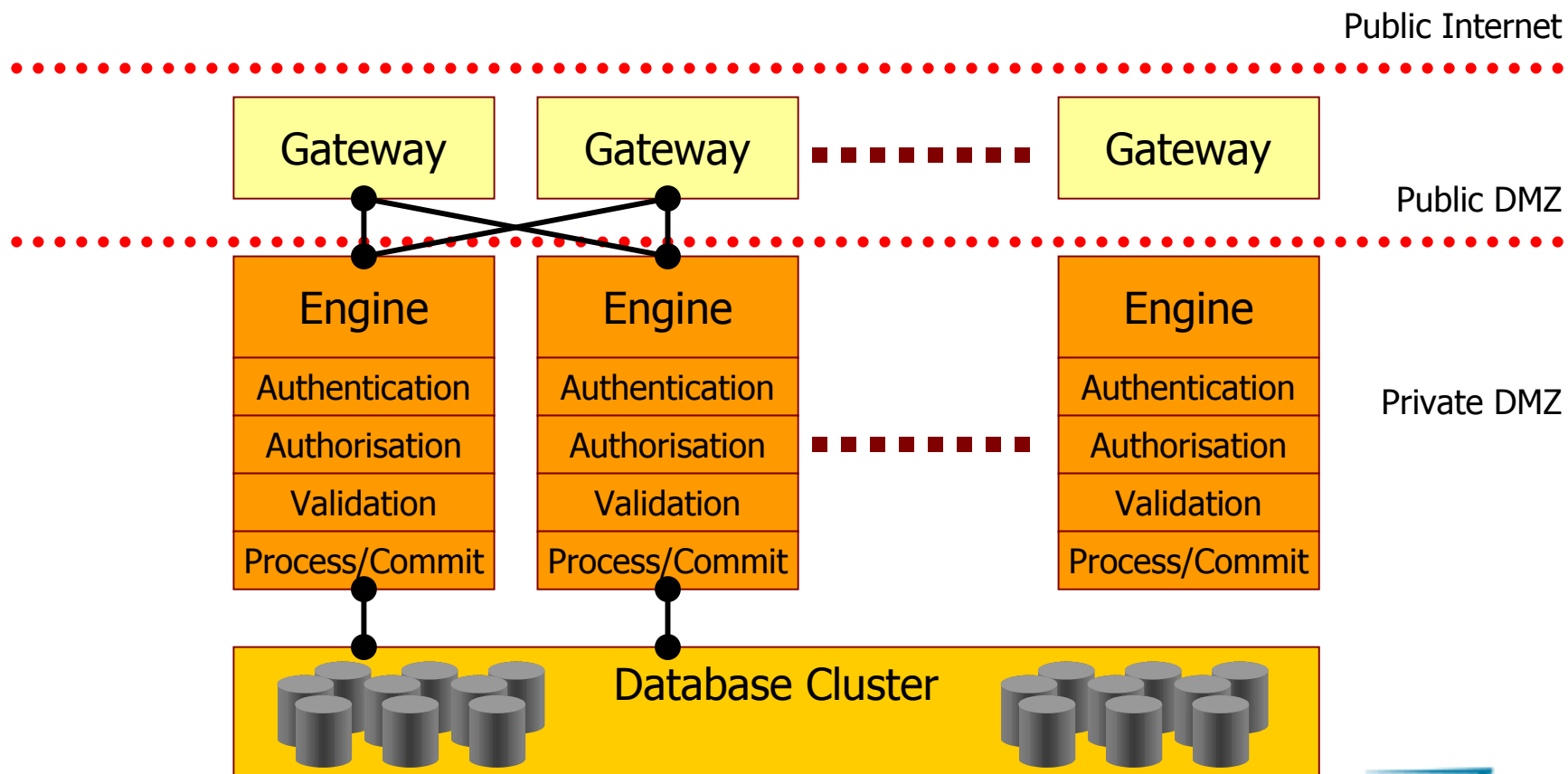
Validation Profiles are stored in the database. Validators are Java class files loaded by the engine as required by a Validator. Validation profiles and Validators are hot swappable

Transactions are loaded as required by incoming request. Cartridges map transactions to legacy interfaces and are loaded according to Database constraints. Specific Instance Data comes from properties files. All components can be added dynamically



Implementation topology ?

WTS is implemented in a clustered configuration for performance, scalability and failure recovery.





The Challenges

- Initially developed for the Microsoft Win/2000 platform
- A Java application... not J2EE based
- Heavily reliant on threads
- Consists of multiple instances of communicating components
- Uses Microsoft SQL/Server database with stored procedures
- Implements SSL libraries at low level
- Inherent failure recovery and scalability
- The “conflict of cultures”



The Methodology

- Phased approach
 - Step 1 - Port application as is
 - Step 2 - Stress the application
 - Step 3 - Modify for Himalaya based on stress testing
 - Repeat steps 2 & 3 until satisfied with performance
 - Step 4 - Benchmark for performance metrics
- Port the most difficult modules first
- Port the Database last
- Get Himalaya support personnel involved from the very start.



Himalaya - pre requisites

The following software is required on the Himalaya

- Operating System: G06.14 or higher
- Database: SQL/MX and/or SQL/MP
- Other: Open System Services (OSS)
JVM version 1.2.2.8
T0083V20_16NOV2001_jdk122_V20AAK
iTP Secure WebServer 5.0
Java Servlets for iTP

Note: Enterprise Java beans were not required for WTS



The experience

- Phase 1 completed in just under 2 weeks (excluding database)
- Database required extensive rework due to table name restrictions and stored procedures
- Encountered JVM problem during initial stress testing
- Encountered early performance restrictions on a number of fronts due to implementation
- Found problems with SIP distributorabend
- Became obvious early in the exercise that we would need to make changes to the application to suit the Himalaya
- Heavy reliance on threads caused performance issues



Himalayarizing the WTS

- Change Database table name formats and remove stored procedures
- Static Vs. dynamic SQL calls
- Introduce SIP to effectively take advantage of the Himalaya fail over/recovery architecture
- Change methodology for handling TCP/IP sockets
- Modification of architecture to reduce emphasis on threads
- Caching - Multiple JVM's in a SIP environment



The result

- Success! Java on a Himalaya works!
- A demonstrable and commercially viable Java based application on the NonStop Himalaya
- Favourable architectural review from Cupertino Advanced Technology Group
- More work required from a performance standpoint
- Great assistance from ~~Tandem~~ ~~Compaq~~ HP support



Special thanks to

- John Turnbull - Compaq Sydney
- Peter Bignell - Himalaya support, Melbourne
- Steve Moriarty - ATG design architect, Cupertino
- Tom Rohner - Java product management, Cupertino
- Thomas Jensen - Himalaya support
- JVM Development team - Cupertino