Introduction to Cassette Development

Lance Bader
Jim Reach
03-16-00

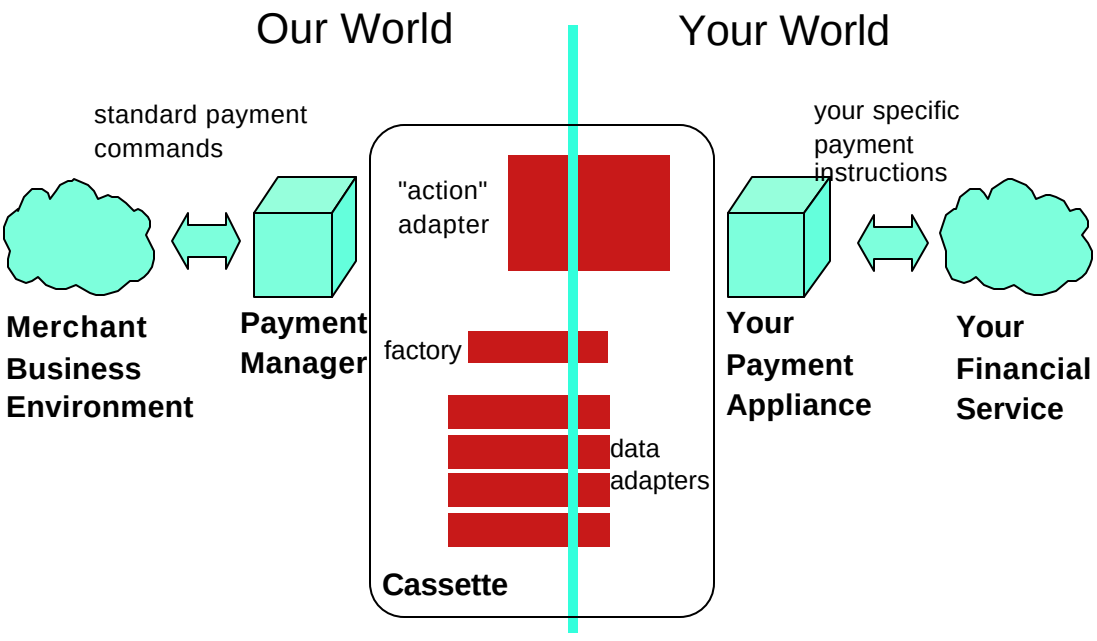
Why do merchants buy IBM Payment Manager?

- Payment Manager provides a consistent interface for diverse payment instruments
  - browser-based user interface
  - programmatic interface
    - for e-commerce catalog system
    - for business back office systems
- Payment Manager supports new/better/cheaper payment instruments without radical change to merchant software
- Payment Manager supports several payment instruments at the same time

Why should you develop an IBM Payment Manager cassette?

- A single work effort integrates your financial service with diverse merchant/business systems via the Payment Manager framework
- Your fine work, based on your quality financial service, can now easily displace others who have chosen this model!

The World View



Bridging the Worlds

- Payment Manager provides a unified merchant view to multiple financial services
  - merchant oriented data objects
  - merchant payment commands
- Your Payment Appliance provides the view to your specific financial service
  - financial system account
  - actions of the financial system
- A Payment Manager Cassette bridges these two views

Implementing the Bridge

- Cassette Account Object
  - holds static data representing a merchant's financial account & access to it
  - has methods that implement actions on that account
- Cassette Data Objects
  - hold the dynamic (per transaction) data necessary for those actions
- Cassette Object
  - a factory used by the Payment Manager for obtaining your Account and Data Objects
- Cassette Query Object
  - an object which provides a strategy for responding to query requests

Our Template Implementation

- Provides most of the logic for common financial service applications
- Your Changes:
  - decide on your cassette name
  - convert template Java files into a cassette skeleton
  - convert the template persistent data to your cassette-dependent data
  - connect the template Account methods to your Payment Appliance methods
  - create a ResourceBundle for national language messages and country-dependent data
  - create an XML document with PSPL tags describing how to display and update your persistent data

Developer Advisory Statement

- We think our template implementation covers many common financial service needs
  - especially variations on the payment card
- We've dropped a huge number of implementation options and opportunities to focus on these common occurrences
- Much of the rest of this presentation covers material that allows you to "step outside the box" if needed