

07-02-02

60393024-062802

A/p10v

06/28/02
 16682 U.S. PRO

PTO/SB/16 (8/96)

Approved for use through 01/31/98. OMB 0651-0037
 Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

Attorney Docket No. 3399P100Z

Type a plus sign (+) inside this box [+]

PROVISIONAL APPLICATION FOR PATENT COVER SHEET

This is a request for filing a PROVISIONAL APPLICATION FOR PATENT under 37 CFR 1.53 (c).

INVENTOR(s)/APPLICANT(s)

LAST NAME	FIRST NAME	MIDDLE NAME/ INITIAL	RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY)
Lund	Tomas	G.	Växjö, Sweden
Kjellberg	Rikard	M.	Stockholm, Sweden
Bergljung	Karl	Martin	Växjö, Sweden

TITLE OF THE INVENTION (280 characters max)

Content Analysis Manager

CORRESPONDENCE ADDRESS (including country if not United States)

Jordan M. Becker

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN, LLP

12400 Wilshire Boulevard, Seventh Floor

Los Angeles, California 90025-1030

Telephone: (408) 720-8300

FAX: (408) 720-8383

ENCLOSED APPLICATION PARTS (check all that apply)

☒ Specification Number of Pages 54 ☐ Small Entity Statement

☐ Drawing(s) Number of Sheets ☒ Other (specify)
Express Mail Certificate

**METHOD OF PAYMENT OF FILING FEES FOR THIS
PROVISIONAL APPLICATION FOR PATENT (check one)**☒ A check or money order is enclosed to cover the filing fees☒ The Commissioner is hereby authorized to charge
filing fees and credit Deposit Account No. 02-2666

Filing Fee Amount (\$) 160.00

This invention was made by an agency of the United States Government or under contract with an agency of the United States Government..

☒ No☐ Yes, the name of the U.S. Government Agency and the Government Contract Number are:

Respectfully submitted,

SIGNATURE



DATE

6/28/02

TYPED or PRINTED NAME: Jordan M. Becker

REGISTRATION NO. 39,602

(if appropriate)

☐ Additional inventors are being named on separately numbered sheets attached hereto

12/01/97

- 1 -

PTO/SB/05 (12/97)

Netflix v. VideoLabs

IPR2023-00628

Netflix. Ex. 1005

EXPRESS MAIL CERTIFICATE OF MAILING

EL821850628US

"Express Mail" mailing label number: _____

Date of Deposit: June 28, 2002

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Julie Arango
(Typed or printed name of person mailing paper or fee)

Julie Arango
(Signature of person mailing paper or fee)

6/28/02
(Date signed)

Serial/Patent No.: _____
Client: Openwave Systems Inc.
Title: Content Analysis Manager

Filing/Issue Date: _____

BSTZ File No.: 3399P100Z
Date Mailed: 6-28-02

Atty/Secty Initials: JMB/jaa
Docket Due Date: ***

The following has been received in the U.S. Patent & Trademark Office on the date stamped hereon:

- | | |
|--|--|
| ☐ Amendment/Response (____ pgs.) | ☒ Express Mail No. <u>EL821850628US</u> ☒ Check No. <u>51057</u> |
| ☐ Appeal Brief (____ pgs.) (in triplicate) | ☐ _____ Month(s) Extension of Time Amt: <u>\$160.00</u> |
| ☐ Application - Utility (____ pgs., with cover and abstract) | ☐ Information Disclosure Statement & PTO-1449 (____ pgs.) ☐ Check No. _____ |
| ☐ Application - Rule 1.53(b) Continuation (____ pgs.) | ☐ Issue Fee Transmittal Amt: _____ |
| ☐ Application - Rule 1.53(b) Divisional (____ pgs.) | ☐ Notice of Appeal |
| ☐ Application - Rule 1.53(b) CIP (____ pgs.) | ☐ Petition for Extension of Time |
| ☐ Application - Rule 1.53(d) CPA Transmittal (____ pgs.) | ☐ Petition for _____ |
| ☐ Application - Design (____ pgs.) | ☒ Postcard |
| ☐ Application - PCT (____ pgs.) | ☐ Power of Attorney (____ pgs.) |
| ☒ Application - Provisional (<u>54</u> pgs.) | ☐ Preliminary Amendment (____ pgs.) |
| ☐ Assignment and Cover Sheet | ☐ Reply Brief (____ pgs.) |
| ☒ Certificate of Mailing | ☐ Response to Notice of Missing Parts |
| ☐ Declaration & POA (____ pgs.) | ☐ Small Entity Declaration for Indep. Inventor/Small Business |
| ☐ Disclosure Docs & Orig & Copy of Inventor's Signed Letter (____ pgs.) | ☒ Transmittal Letter, in duplicate |
| ☐ Drawings: _____ # of sheets includes _____ figures | ☐ Fee Transmittal, in duplicate |

☐ Other: _____

PROVISIONAL UNITED STATES PATENT APPLICATION

Content Analysis Manager

INVENTORS:

Tomas G. Lund
Rikard M. Kjellberg
Karl Martin Bergljung

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026
(408) 720-8300

Docket No. 3399P100Z

"Express Mail" mailing label number: EL821850628US

Date of Deposit: June 28, 2002

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Julie Arango
(Typed or printed name of person mailing paper or fee)

Julie Arango
(Signature of person mailing paper or fee)

6/28/02
(Date signed)

Openwave Systems Inc.
Inventor Name: Tomas Lund

Group Name:

IDEA PROPOSAL

Problem

An essential part of a supply chain management workflow is to do a quality check in the supplied product. This quality check can either just be a check that the product fulfills prescribed quality parameters or it could invoke a process to modify the product so that the quality parameters are fulfilled.

The processing of a product depends on properties of the product and on the intended usage of the product. The problem that we are trying to solve is *how to apply a generic analysis process on many different content types and many different intended uses of the content*.

Furthermore, the process must support both a real-time process, where the content is analyzed just prior to being delivered to a recipient, and an off-line process, where content is analyzed prior to being made available to recipients.

Known approaches

One typical approach is to manually apply one processing/checking-operation at the time. Another way is to design specific analysis mechanisms for each particular content type and its intended use.

Comments

This solution is preferred over others because:

- It can be used in the supply chain, where the real-time aspect is non-critical.
- It can be used in the delivery chain, where the real-time aspect is absolutely critical.
- The workflow applies to all content types for all purposes.
- Functionality can be assembled for a wide variety of uses.
- Promotes re-use, since *processors* can be re-used in many different *scenario configurations*.

From what we have seen so far, this approach to content analysis AND its position within the overall system (see below), has given us a very powerful mechanism to introduce new features related to product supply and product delivery.

The CAM is invoked in two different workflows:

1. During product supply, where a supplier submits content for analysis and approval. Once approved, the content will be made available to end-users. This is the off-line aspect of the CAM.
2. During product delivery, where end-users request a list of available content, selects content, negotiate a price and has it delivered to a user device. This is the real-time aspect of the CAM.

A product processor defines one type of quality operation onto an abstract representation of the product. A number of product processors are grouped together into a quality process configuration. Based on the quality and delivery constraints the system locates a quality process configuration and applies the products processors to the product. First, it applies those processors that do a non-destructive check of the product. If any of these processors indicates that the product to not fulfill the quality/delivery constraints the process is terminated and the product is rejected. Otherwise the potentially modifying processors are applied to the product in a configured order. The result of the process is an approval of the product (initial or modified).

We start of defining the process workflow, which is independent of content type and its intended use. The workflow contains the following steps (or phases):

1. Inspection
2. Preparation
3. Analysis

4. Approval
5. Packaging

(See details in section 2.2 of MAP-002-FS-11).

Step 4 and 5 are generally swapped, so that step 5 occurs before step 4.

The Content Analysis Manager (CAM) has the following components:

1. **Phases**, which define a set of possible analysis actions in a certain part of the analysis workflow.
2. **Processors** (called Inspectors in MAP-002-FS-11), which perform some isolated analysis task in a certain phase.
3. **Scenario configurations** (or simply configurations), which combine a number of processors into a complete CAM workflow.

Ellipsus Systems

infiniteMAP Feature Specification Content Analysis Manager

Version 1.0

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

Revision History

Date	Version	Description	Author
	0.1	Setup	Tomas Lund
	0.2	Additional descriptions	Rikard Kjellberg
	0.3	Filled out the feature set definitions	Tomas Lund
	0.4	Continued to expand on features	Tomas Lund
	1.0	Ratified	Martin Bergljung

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

Table of Contents

1.	Introduction	4
1.1	Purpose	4
1.2	Scope	4
1.3	Definitions, Acronyms and Abbreviations	4
1.3.1	Definitions	4
1.3.2	Acronyms	4
1.3.3	Abbreviations	4
1.4	References	5
1.5	Overview	5
2.	System Description	6
2.1	Purpose	6
2.2	Overview	6
2.2.1	Inspection	6
2.2.2	Preparation	7
2.2.3	Analysis	7
2.2.4	Approval	7
2.2.5	Packaging	9
2.3	Logical View	9
2.4	Use-Case view of the Content Analysis Manager	9
3.	Features	10
3.1	Requirements	10
3.2	Feature Set: Automatic approval of an implementation	11
3.3	Feature Set: Manual approval of an implementation	11
3.4	Feature Set: Managing the processors of a content analysis	12
3.5	Feature Set: Inspecting the outcome of the analysis of an implementation	13
3.6	Feature Set: Downloading an implementation	13
3.7	Feature Set: Inspecting an implementation	13
3.8	Feature Set: Preparing an implementation during deployment	13
3.9	Feature Set: Preparing an implementation during provisioning	13
4.	Architecturally Significant Design Packages	15

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

Feature Specification

1. Introduction

1.1 Purpose

This document describes the detailed features for the Content Analysis Manager (CAM), which is called a *major feature set* (MFS). It uses the requirements from the functional requirements specification [R.1] as *feature sets* (FS) and details them into features.

1.2 Scope

1.3 Definitions, Acronyms and Abbreviations

The definitions, acronyms and abbreviations from [A.1] apply.

1.3.1 Definitions

Term	Description
Product	A product consists of a product description and the implementations of the product.
Implementation	The implementation of a product consists of two parts – the description and its binaries. The implementation description is typically downloaded to the client and drives the downloading of the binaries of the implementation.
Binary	A binary is the content delivered to the client that constitutes the product.
Content	Content is a synonym for description of the implementation and all binaries that belong to an implementation.
Instrument	Modify the object code in a binary so that additional code gets executed. Patching is often used as a synonym to instrumentation.

1.3.2 Acronyms

Acronym	Description
MFS	Major Feature Set
FS	Feature Set
SLA	Service Level Agreement
CAM	Content Analysis Manager
MAP	Mobile Application Provisioning system

1.3.3 Abbreviations

Abbreviation	Description
Pri	Priority

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

1.4 References

Id	Description (title, version, date, publisher, source)
[R.1]	Functional Requirement Specification, MAP-002-RS-01, Ellipsus Systems.

1.5 Overview

This FS is a detailed design document derived from applicable requirements in [R.1], which is the overall functional requirement specification of the MAP.

Each feature is assigned a priority (pri). This priority can take the following values:

Value	Priority	Meaning
L	Low	This feature can be deferred to later in the project or even to a later release.
M	Medium	This feature can be deferred to later in the project but should be implemented prior to first release.
H	High	This feature should be implemented as soon as possible to verify that it can be realized, or to be able to implement other aspects of the system, which rely on this requirement.
N/A	Not Applicable	This feature is not subject to prioritization since it is of general character and affects the overall development project.

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

2. System Description

2.1 Purpose

The primary purpose of the Content Analysis Manager is to provide maximum user experience by ensuring the highest *content quality* possible. This means that the Content Analysis Manager shall guarantee that content (i.e. binaries and descriptors) can be delivered to, and executed on a client device in accordance with criteria defined by the operator of the provisioning system.

2.2 Overview

The products that are managed by *infiniteMAP* are of many different types. In principle, the implementations and the corresponding content could be treated as just 'black-boxes' (or BLOBs) administered by the system and delivered to the client just as they are. But there are a number of functions related to the content that could be applied in order to improve the user experience of using *infiniteMAP* for product delivery. The Content Analysis Manager is the grouping of all such functions within *infiniteMAP*.

Content analysis is performed in steps, or sub-processes. In each of these steps, a number of analysis functions can be applied depending both on the type of the content and on the need of the system:

1. Inspection
2. Preparation
3. Analysis
4. Approval
5. Packaging

2.2.1 Inspection

The content is inspected for compliance with its declared type. The Content Analysis Manager accepts *Inspector plug-ins* for this purpose. Each content type can use one or more Inspectors. An Inspector is typically associated with one content type, but could work on several content types. Should a content type have a list of Inspectors, each Inspector is called in sequence until an inspection fails or all inspections pass.

Different contents need different types of inspectors. Examples of inspectors:

Content type	Inspector
MIDlet jar	• Malicious content (could be performed as an external anti-virus service check) • Pre-verified check • API restrictions check
MIDlet descriptor	• Correspondence between descriptor and binary
Ringtone	• Proper format and encoding
Sound	• Proper format and encoding
Applet jar	• Malicious content (could be performed as an external anti-virus service check) • API restrictions check

An Inspector will only examine content and indicate the result of this process. It will NOT perform any modification or remedy to content.

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

2.2.2 Preparation

Preparation is the process of altering content so that it conforms to the environment where it will be published and made available to the users. The content needs to be prepared both based on its content type and on how it should be handled by the system. Examples of preparations that need/can be applied:

- **Descriptor re-write.** Descriptor files must be re-written to follow the policies of the system with regard to location of binaries, downloading procedures (e.g. OTA downloading), device type etc.
- **Instrumentation.** Executable can possibly be instrumented so that code required for proper functionality of the system gets executed on the device. Time-limited or demo software, special pricing models, dynamic updating of software, uploading of program data to a server and downloading of data to the device are examples of functions that can be obtained by instrumentation.
- **Pre-verification.** Should a J2ME application not already be pre-verified, it will be performed in the preparation stage.
- **Conversions.** Media such as images, ring-tones or sounds might need to be prepared for conversion to different formats in order to accommodate a wider range of devices.
- **Manifest re-write.** The manifest is stored in the jar-file and could need to be modified.

In those cases where content resides outside of the provisioning system, some of the preparation may need to be re-invoked in real-time (depends on service level agreement - SLA). This is the case with instrumentation and pre-verification, for example.

2.2.3 Analysis

In order to be able to deduce as much information as possible from the content it needs to be analyzed. For Java code this involves an examination of which classes/methods/APIs are used by the code. The Content Analysis Manager accepts analysis plug-ins, associated with content type.

The result of an analysis is a readable report (XML?), which can be used during content approval.

2.2.4 Approval

The approval process is a manual and/or automated process (depends on SLA) where deployed content is approved for publishing to the consumers. The approval process can be categorized as follows:

- **Approval configuration**, where the approval process is configured.
- **Automated approval actions**, where certain aspects are automatically checked and approved.
- **Manual approval actions**, where certain aspects are manually checked and approved.

2.2.4.1 Approval configuration

The approval process is configured in various ways. Examples of configurations are:

- Setting up API-call restrictions.
- Setting up available instrumentations.

2.2.4.2 Automated approval actions

The approval process can perform a number of automated approvals. Examples are:

- Checking that restricted API-calls are absent from the content.

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

- Checking that pre-verification was successful.
- Checking that approval has been performed in accordance with SLA.

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

2.2.4.3 Manual approval actions

The approval process can allow a number of manual approvals. Examples are:

- Examination and approval of analysis result.
- Dry-running an application in a browser-based emulator.
- Approve applied or suggested instrumentations.

2.2.5 Packaging

Before delivering content to the client device it will be packaged in a suitable format. Packaging can include functions for:

- Packing/re-packing several MIDlets together in one jar-file.
- Calculating a checksum and signing the content.
- Adding headers and tails appropriate for delivery mechanism and device (in the case of SMS delivery, for example).

All of these actions may be performed in real-time.

The package mechanism is tightly coupled with content type and device type. The system accepts packaging plug-ins, which are associated with content type.

2.3 Logical View

TBD

2.4 Use-Case view of the Content Analysis Manager

The following use-cases describe the role of the Content Analysis Manager.

Actor	Use-case	Description
Consumer	Downloading	The content that the consumer downloads should function properly on the device onto which the consumer downloaded the content.
Product admin	Approving content	Verify that the content is appropriate for downloading
Product provider	Assigning required capabilities	Assign the device capabilities that this product requires to be present on the device.
Product admin	Allowable content	Administer the allowed contents and the allowed usage by the content
Product admin	Packaging	Administer how content can be packaged and re-packaged.
Product provider	Content integrity	Ensure the integrity of approved content.

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

3. Features

3.1 Requirements

The following requirements have been extracted from [R.1] and form the *feature sets* for this module. One requirement can result in several Feature Sets. Several requirements can also fit in one single feature set.

Req #	Requirement	Pri	Description	Notes	Owner	Feature Set
3	Deployment support		It shall be possible to check products for malicious content upon deployment into the MAP.			
4	Configuration		It shall be possible to add and remove malicious product patterns in the system.			
116	Approval		All submitted products shall be approved prior to making it available to the consumers.			
117	Approval by contract		It shall be possible to set up a contract between a product provider and the MAP, so that approvals are not necessary.			
118	Examination		For the purpose of approval, the administrator shall be able to test and examine submitted products.			
122	Product verification		The MAP shall contain an automated pre-verification process for all products.			

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

3.2 Feature Set: Automatically Approving an Implementation

When all parts of an implementation have been deployed the system initiates the process of analyzing the contents of the implementation. If the product provider is a trusted provider this process tries to do an automatic approval of the implementation based on the criteria that the product administrator have defined. If the implementation passes the approval process then the implementation is marked as approved. If it is not approved the implementation is marked as not approved and the product administrator must perform a manual approval process.

#	Feature	Notes	Pri	Owner
1	Determine the <i>set of processors</i> for an approval process		H	
2	Perform the <i>inspection</i> of an implementation		H	
3	Perform the <i>preparation</i> of an implementation		H	
4	Perform the <i>analysis</i> of an implementation		L	
5	Perform the <i>approval</i> of an implementation		M	
6	Perform the <i>packaging</i> of an implementation		M	
7	Store the <i>analysis result</i> for an approval process		H	

3.3 Feature Set: Manually Approving of an Implementation

When all parts of an implementation have been deployed and the deployer is not a trusted product provider the system initiates an initial content analysis of the implementation (steps 1 to 4 of the 3.2

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

Feature Set: Automatically Approving an Implementation). The product administrator, who can either approve the implementation or remove it, inspects the result of the analysis.

#	Feature	Notes	Pri	Owner
1	Retrieve the <i>analysis result</i> for an implementation approval process	If no result exists then rerun the FS Automatic approval	L	
2	Assign the <i>approval status</i> for an implementation		H	
3	Configure the <i>data</i> for an implementation	Perform any configuration tasks on the implementation to make it 'approvable'	M	

3.4 Feature Set: Managing the Processors of a Content Analysis

The product administrator manages the configuration of processors by adding, removing and configuring individual processors. The product administrator can also manage other types of configurations such as allowed API calls, allowed packages, allowed OEM APIs, instrumentations, etc.

#	Feature	Notes	Pri	Owner
1	Add the <i>processor</i> for a content type		H	
2	Remove the <i>processor</i> for a content type		H	
3	Assemble the <i>processors</i> for a configuration		H	

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

3.5 Feature Set: Inspecting the Outcome of the Analysis of an Implementation

The product administrator or the product provider can inspect, and possibly act on, the outcome of the analysis.

#	Feature	Notes	Pri	Owner
1	Modify the <i>required capabilities</i> for an implementation		H	

3.6 Feature Set: Downloading an Implementation

When the consumer downloads an implementation the content, descriptor and/or binaries, might need to be prepared for the proper dynamic delivery of the content.

#	Feature	Notes	Pri	Owner
1	Select the <i>processor configuration</i> for provisioning		H	
2	Prepare the <i>implementation</i> for provisioning		H	

3.7 Feature Set: Inspecting an Implementation

The inspection of an implementation applies a set of configured inspectors (processors) to the implementation.

#	Feature	Notes	Pri	Owner
1	Inspect the <i>JAD descriptor</i> for conformance with specification		H	
2	Inspect the <i>MIDlet jar file</i> for correctness		H	
3	Inspect the <i>JAR-file</i> for viruses.		H	
4	Inspect the <i>MIDlet-jar file</i> for fulfilling pre-verification checks		H	
5	Inspect the <i>JAR file</i> for usage of disallowed APIs		H	

3.8 Feature Set: Preparing an Implementation during Deployment

The preparing step for deployment applies a set of configured preparators (processors) to the implementation.

#	Feature	Notes	Pri	Owner
1	Rewrite the <i>MIDlet jad-file</i> for an implementation		H	
2	Rewrite the <i>Manifest</i> of a MIDlet jar-file		H	
3	Rewrite the <i>Applet html-file</i> for an implementation		H	
4	Rewrite the <i>WebStart jnlp-file</i> for an implementation		H	
5	Apply the <i>pricing model instrumentation</i> to an implementation		M	

3.9 Feature Set: Preparing an Implementation during Provisioning

The preparing step for provisioning applies a set of configured preparators (processors) to the implementation.

#	Feature	Notes	Pri	Owner
1	Rewrite the <i>MIDlet jad-file</i> for provisioning of an implementation.		H	
2	Rewrite the <i>Applet html-file</i> for provisioning of an implementation.		H	

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

3	Rewrite the <i>WebStart jnl-file</i> for provisioning of an implementation.		H	
---	---	--	---	--

infiniteMAP	Version: 0.4
Feature Specification	Date:
MAP-002-FS-11-ContentAnalysisMgr	

4. Architecturally Significant Design Packages

The Content Analysis Manager is implemented by classes and interfaces in the following packages.

- `com.ellipsus.imap.content.analysis` – Interfaces for the Content Analysis Manager.
- `com.ellipsus.imap.content.analysis.ejb` – The EJBs implementing the manager.
- `com.ellipsus.imap.content.analysis.servlet` – Classes to manage the manager from the web.
- `com.ellipsus.imap.content.analysis.adapter` – Implementation of standard processors
- `com.ellipsus.imap.content.analysis.util` – Utilities for standard processors
- `com.ellipsus.imap.content.analysis.vo`



60393024 . 062802

Content Analysis Management

Technical overview

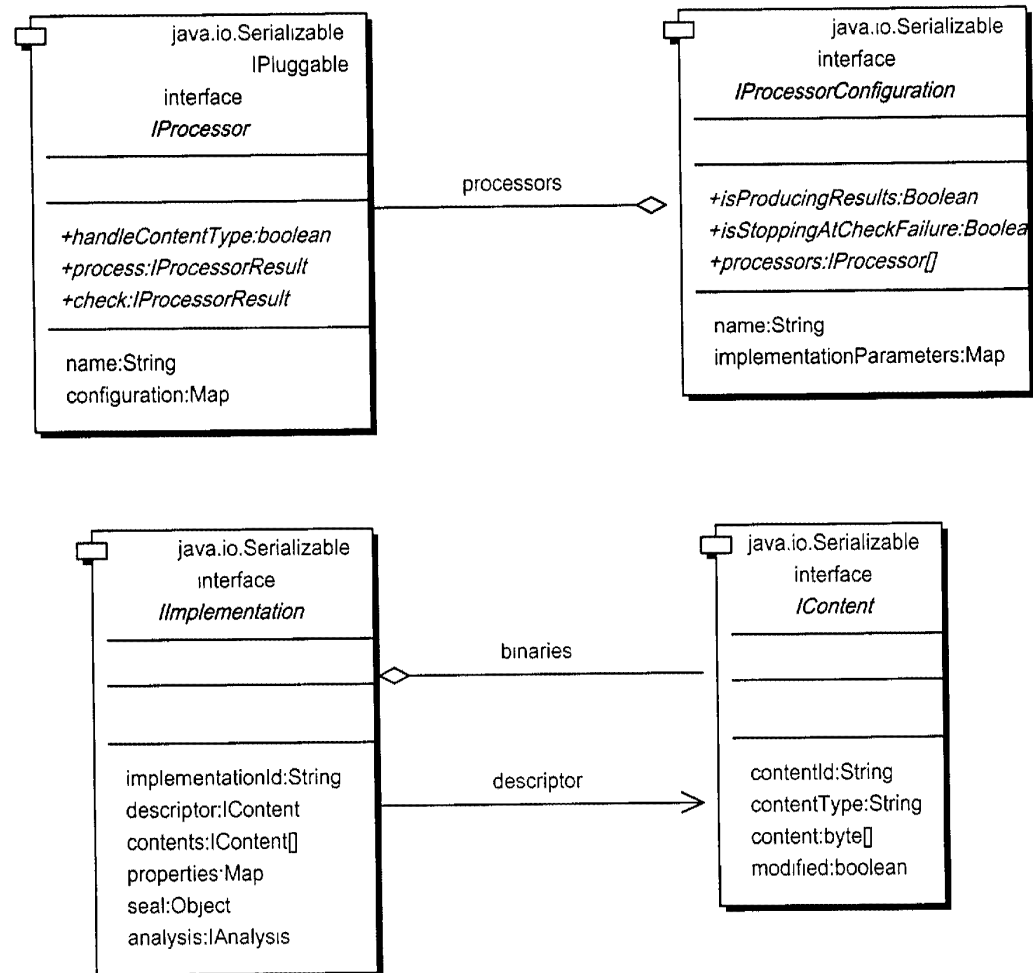
Agenda

Explore the building blocks of the Content Analysis Manager

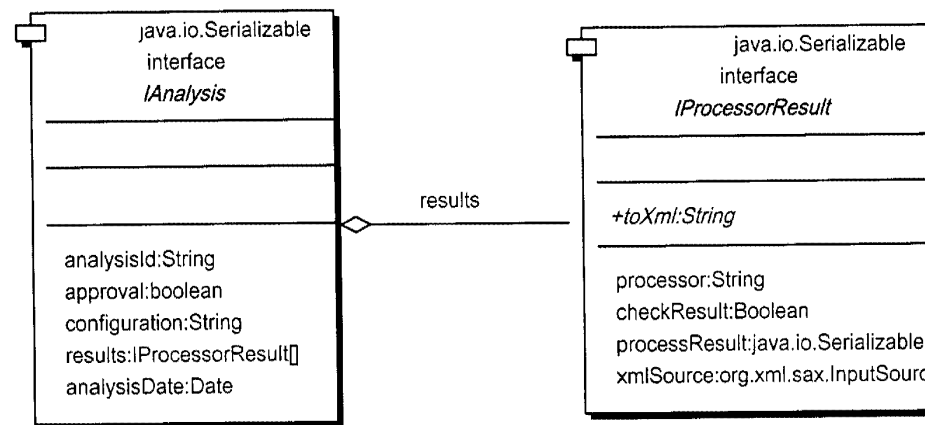
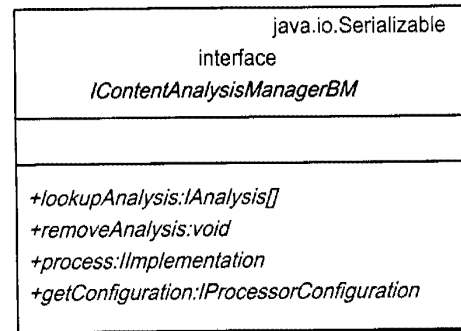
Explore the interface to the Content Analysis Manager

Examples of implementations of processors

Building blocks - UML View



Building blocks – UML View (cont.)



Building blocks

- IProcessor – The work horse
- IImplementation – The thing to analyze
- IContent – The content of the implementation
- IProcessorConfiguration – A collection of processors to use for an analysis.
- IAnalysis – The result from an analysis
- IProcessorResult – The result from one processor

Building blocks - IProcessor

- A processor is the class performing a specific analysis task.
- A processor can decide if it wants/can process specific content
 - *boolean handleContentType(String contentType);*
- A processor can check the content – no modification to the content.
 - *IProcessorResult check(IImplementation content)*
- And process the content – possible modification of the content
 - *IProcessorResult process(IImplementation content)*

Building blocks - IImplementation

- An Implementation consists of one IContent that describes the implementation (the descriptor) and 0 or more IContent objects (the 'binaries').
- Methods
 - *String getId();*
 - *IContent getDescriptor();*
 - *IContent[] getContents();*
 - *Map getProperties();*
 - *Object getSeal();*
 - *IAnalysis getAnalysis();*

Building blocks - IContent

- An IContent describes the smallest unit of content that can be handled by the Content Analysis Manager.
- Its content type describes how the content should be interpreted.
- Methods
 - *String getContentId();*
 - *String getContentType();*
 - *byte[] getContent();*
 - *boolean getModified();*

Building blocks - IProcessorConfiguration

- A processor configuration encapsulates a set of IProcessors. A configuration is setup for different tasks or combinations of processors.
- Methods
 - *String getName();*
 - *Boolean isProducingResults();*
 - *Boolean isStoppingAtCheckFailure();*
 - *Map getImplementationParameters();*
 - *IProcessor[] processor();*

-

Building blocks - IProcessorResult

- The result from a specific IProcessor
- Methods
 - *String* `getProcessor();`
 - *Boolean* `getCheckResult();`
 - *java.io.Serializable* `getProcessResult();`

The managers

- One administrative view of the CAM
 - *Administer processors and configurations*
 - *Administer analysis results*
 - *Administer persistent services for the CAM (e.g. known APIs, instrumentations)*
- One runtime view of the CAM
 - *Apply a configuration to an implementation*
 - *Inspect the outcome of the analysis*

The managers – cont.

- Part of a Content Approval work flow
- Users of the CAM can
 - *Associate processor configurations with suppliers*
 - *Associate processor configurations with characteristics of devices*
 - *Associate processor configurations with dedicated tasks – specific instrumentations, pre-verification runs etc.*
- Users of the CAM should
 - *Provide visual presentation of results*
 - *Provide functions for approval or rejection of implementations.*

Examples

- Two examples of processors (inspect phase)
 - *Virus Inspector*
 - *JAD verifier*
- One example of a non-modifying processor (package-phase)
 - *Jar signer*
- One example of a modifying processor (prepare-phase)
 - *JAD rewriter*

Example – Virus Inspector

```
public class VirusInspector implements IProcessor {
    public VirusInspector() {this("Virus Inspector");}

    public boolean handleContentType(String contentType){return true;}

    public IProcessorResult process(IImplementation content) throws ProcessingException {
        return null; // Nothing to do
    }

    public IProcessorResult check(IImplementation content) throws ProcessingException {
        IContent[] contents=content.getContents();
        ProcessorResultImpl result=new ProcessorResultImpl(getName(),Boolean.TRUE);
        if (contents==null) {
            result.setProcessResult("No binary content");result.setCheckResult(Boolean.FALSE);
            return result;
        }
        for(int i=0;i<contents.length;i++) {
            if (contents[i].getContent()!=null) {
                virusCheck(contents[i].getContent(),errors);
            }
        }
        if (errors.length()>0) {
            result.setProcessResult(errors.toString());result.setCheckResult(Boolean.FALSE);
        }
        return result;
    }
}
```

Example – JAD Verifier

```
public class JadChecker implements IProcessor {
    public JadChecker() {this("Jad Checker");}

    public boolean handleContentType(String contentType) {
        return "text/vnd.sun.j2me.app-descriptor".equals(contentType);
    }

    public IProcessorResult process(IImplementation content) throws ProcessingException {
        return null;
    }

    public IProcessorResult check(IImplementation content) throws ProcessingException {
        // Check properties
        ProcessorResultImpl result=new ProcessorResultImpl(getName(),Boolean.TRUE);
        IContent descriptor=content.getDescriptor();
        String errorMessage=checkDescriptor(descriptor,propMap);
        if (errorMessage!=null) {
            result.setProcessResult(errorMessage); result.setCheckResult(Boolean.FALSE);
        }
        return result;
    }

    private String checkDescriptor(IContent descriptor,Map propMap)
        throws ProcessingException {
        // Read descriptor and check correctness/completeness
    }
}
```

Example – Implementation signer

```
public class JarSigner implements IProcessor {
    private MessageDigest m_digester;

    public JarSigner() {this("Jar Signer");}

    public JarSigner(String name) {
        super(name);
        try {
            m_digester=MessageDigest.getInstance("SHA");
        } catch (NoSuchAlgorithmException nex) {
        }
    }

    public IProcessorResult process(IImplementation content) throws ProcessingException {
        return null;
    }

    public IProcessorResult process(IImplementation impl) throws ProcessingException {
        impl.setSeal(calculateSeal(impl));
        return null;
    }

    private String calculateSeal(IImplementation impl) {
        return base64encode(digester(impl));
    }
}
```

Example – Implementation signer (cont.)

```
private byte[] digest(IImplementation impl) {
    try {
        MessageDigest digest=(MessageDigest)m_digester.clone();
        digest.reset();

        if (impl!=null) {
            if (impl.getDescriptor()!=null &&
                impl.getDescriptor().getContent()!=null) {
                digest.update(impl.getDescriptor().getContent());
            }
            IContent[] bins=impl.getContents();
            if (bins!=null) {
                for(int b=0;b<bins.length;b++) {
                    if (bins[b].getContent()!=null) {
                        digest.update(bins[b].getContent());
                    }
                }
            }
        }
        return digest.digest();
    } catch (Exception ex) {
    }
    return new byte[20];
}
```

Example – JAD Rewriter

- Add implementation of process in the previous example

```
public IProcessorResult process(IImplementation content) throws ProcessingException {
    Map propMap=content.getProperties();
    String baseUrl=(String)"http://";
    if (propMap!=null) {
        if (propMap.containsKey("baseUrl")) {
            baseUrl=(String)propMap.get("baseUrl");
        }
    }
    IContent descriptor=content.getDescriptor();
    String desc=getDescriptor(descriptor.getContent(),null);
    if (m_replaceRe.getMatch(desc)!=null) {
        desc=m_replaceRe.substituteAll(desc,baseUrl);
        descriptor.setContent(getDescriptor(desc,null));
        descriptor.setModified(true);
    }
    return new ProcessorResultImpl(getName(),Boolean.TRUE);
}
```



Content Analysis Management Quality Assurance for 3G Download Vault

Purpose

- Provide maximum user experience by ensuring the highest content quality possible
- Automate the quality control process
- Provide more tools for the developer community

Ensure that content can

- Be delivered to the desired device
- Safely executed on the device
- Executed in a way determined by the operator to give the best user experience

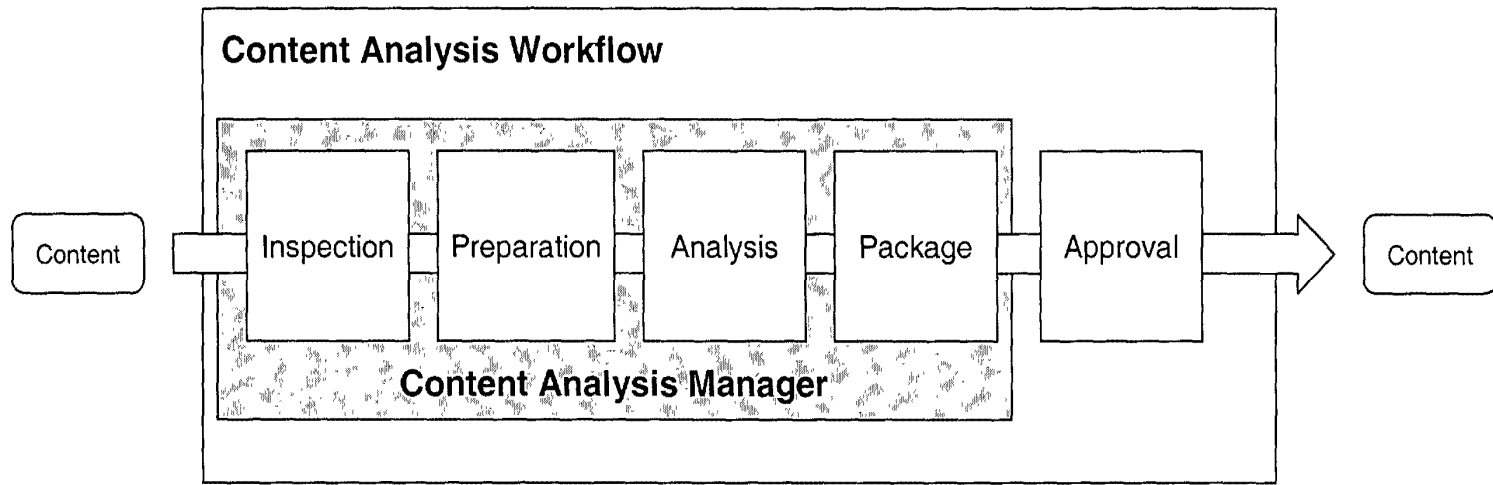
Overview

Content can be of many different kinds:

- J2ME/MIDP applications
- Ring tones
- Wall paper
- Images (personal as well as commercial)
- Descriptors (JAD, GCD)
- Other e.g. URI/URLs

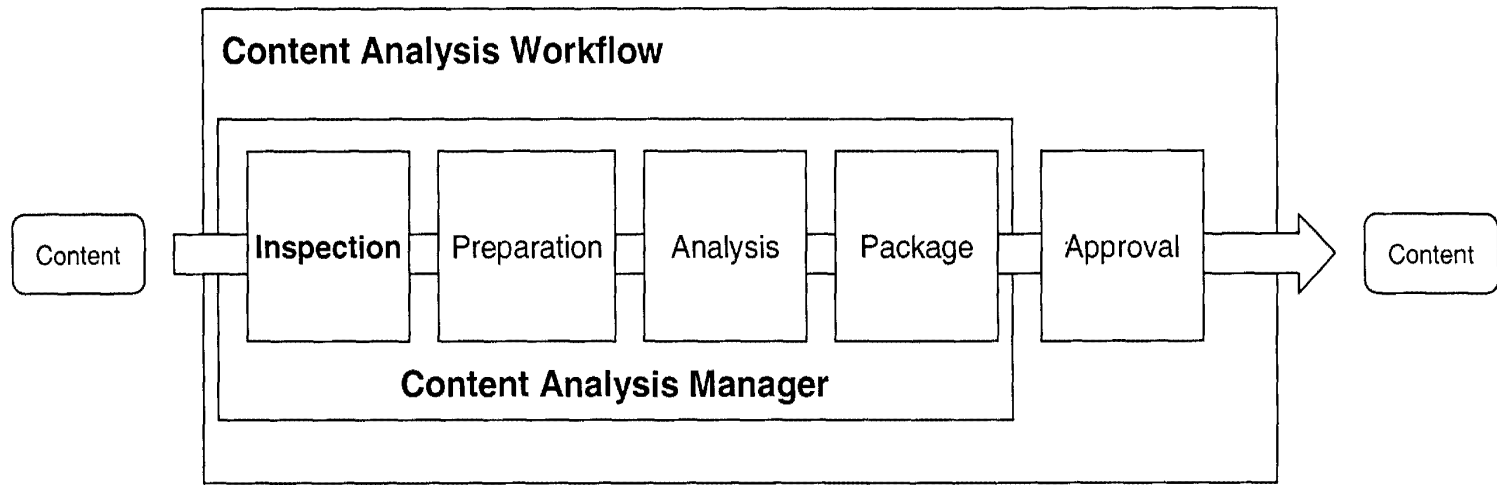
Any content is subject to analysis in order to ensure its quality.

Process



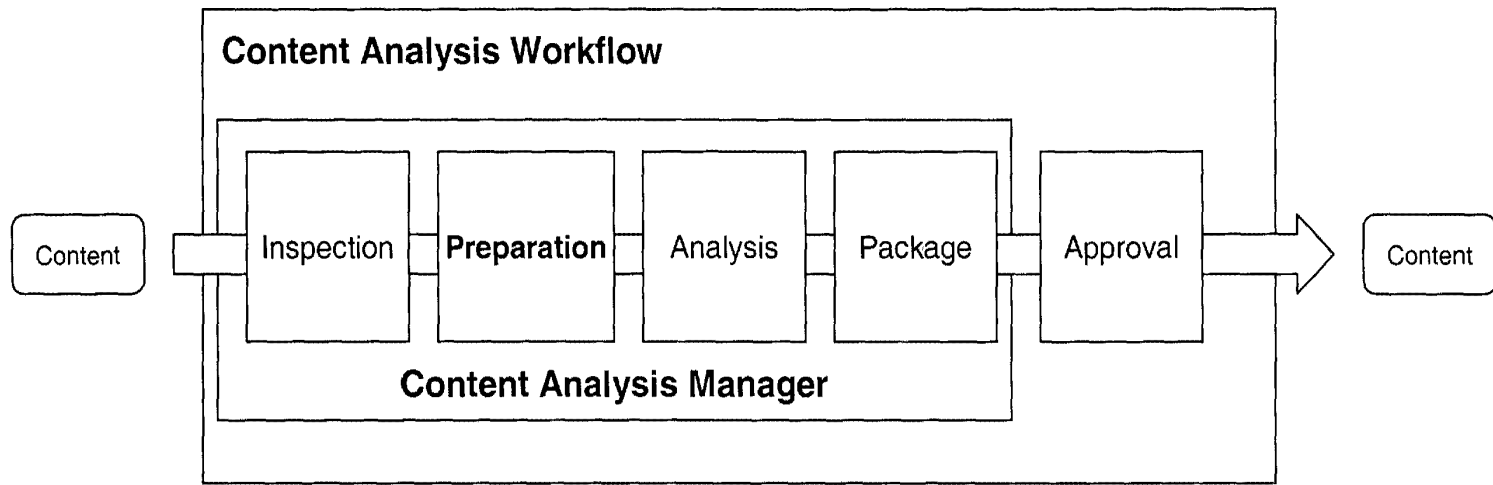
- Supports stocking workflow
- Supports delivery workflow (real-time)
- Central and distributed storage models are supported
- Defines 5 analysis phases – Inspect, Prepare, Analyze, Package and Approve

Inspection Phase



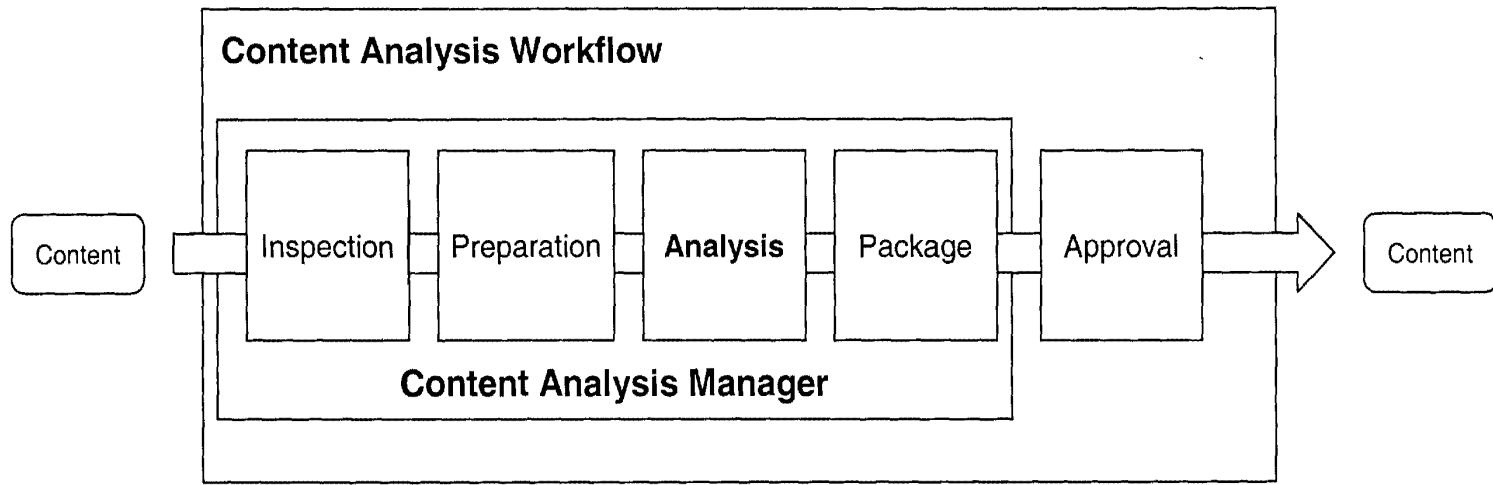
- The content is untouched in the inspection phase.
- This is a read only operation, resulting in a pass/fail response
- Inspects content for
 - ✓ Malicious content (executable code)
 - o Proper encoding (multi-media content)
 - ✓ Pre-verify check (J2ME/MIDP)
 - ✓ API restrictions (Java)
 - ✓ Allow external anti-virus checkers

Preparation Phase



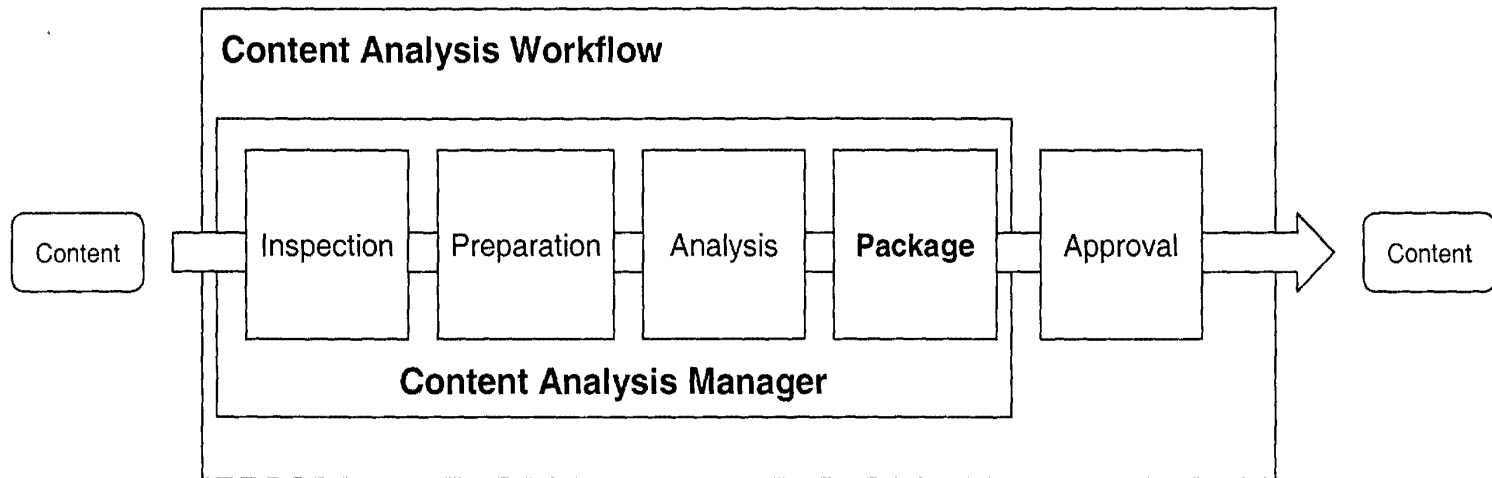
- In the Preparation phase content is altered so that it conforms to the provisioning environment
- Examples of alterations are
 - ✓ Descriptor re-write - both for stocking and delivery, JAD, JNLP, GCD, HTML
 - ✓ Instrumentation – Java technology for DRM support
 - o Pre-verification (J2ME/MIDP)
 - o Conversions (Multi-media)

Analysis Phase



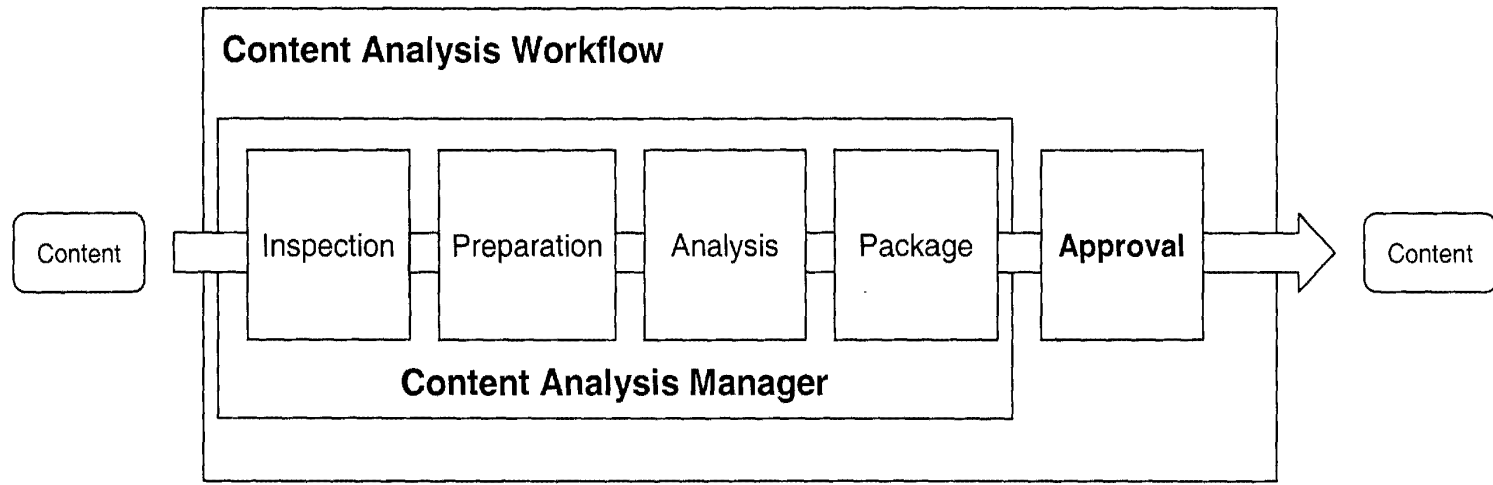
- The Analysis phases performs an examination of the content and varies dependent on content type
- Results in a readable report (XML)
- Examples of analysis aspects are
 - ✓ Java program structure
 - ✓ Used API calls
 - o Resource utilization (heap etc)
 - ✓ Used instrumentations

Package Phase



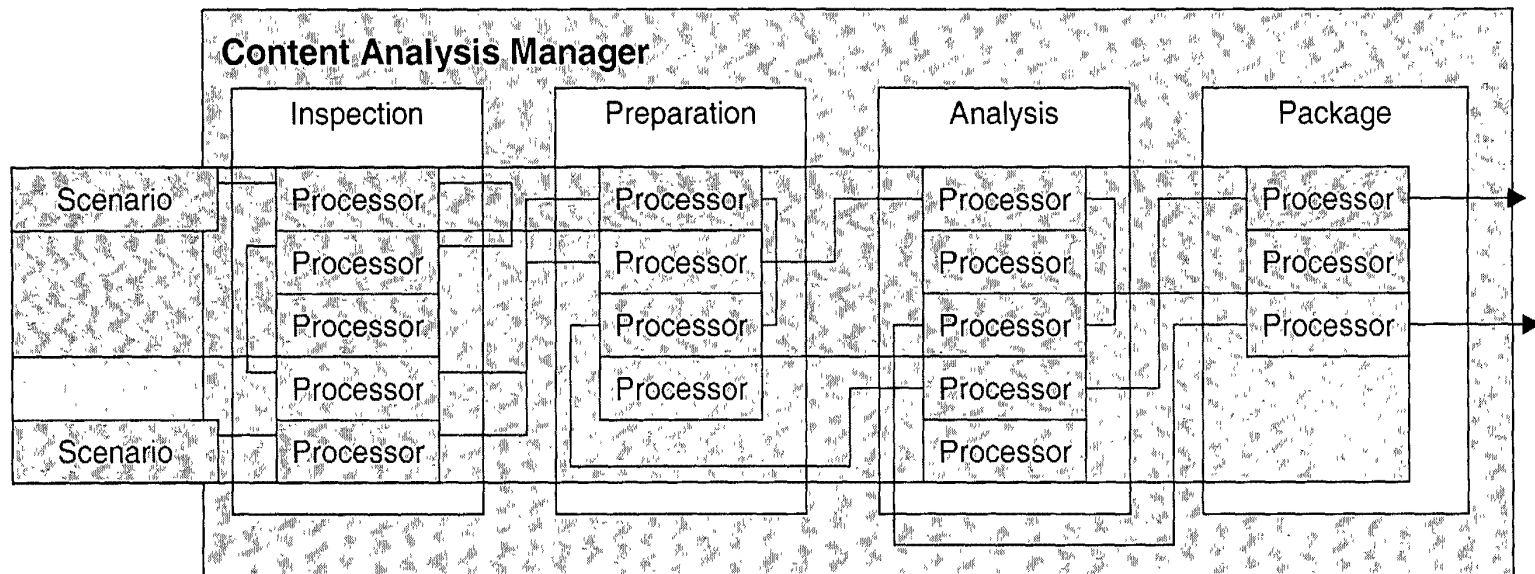
- Packaging content in a format that can be checked for integrity and digested by the targeted device types
- Examples of packaging are
 - ✓ Packing/unpacking midlets in jar-files
 - ✓ Calculating checksum and signing content for integrity
 - o Adding device specific headers and tails for content delivery
 - ✓ Adding authentication key to content and/or descriptors

Approval Phase



- Manual approval based on CAM results (analysis report)
- Can be bypassed if the service level agreement allows that (trusted suppliers)
- Examples of approval activities are
 - Examining the analysis report
 - Dry-running applications in emulator environment
 - Approve the content for publishing

Architecture



Scenario configurations

- Describes a chain of processors
- Based on considerations such as
 - Content type
 - Supplier status (trust level, assigned developer program etc)
 - Destined device type
 - Stocking
 - Delivery

Example – Implementing DRM within CAM

Example on how to implement DRM for J2ME applications

- DRM Model
 - *Protected Content* – content only usable with license
 - *License* – usage rules and keys to unlock content
 - *Delivery* – Matching content with license to user
- CAM Strategies
 - *Preparator to tie MIDlet implementation to license*
 - Instrumentation internally
 - Instrumentation externally
 - Device support for decryption
 - *Packaging to match deliverable with user*
 - Match provisioning method with payment for license

Example – Implementing DRM within CAM (cont.)

- Instrumentation internally
 - *Sub-class all MIDlets with new version that honors the license properties encoded in the JAD*
 - Handle limits on usage (number runs, number of days)
- Instrumentation externally
 - *Let external tool prepare descriptor and MIDlet jar*
- Device support
 - *Encrypt content*
 - *Or calculate MAC/Hash for content*

Example – Implementing DRM within CAM (cont.)

- Packaging
 - *Modify JAD to include authentication token – client only solution.*
 - *Modify JAD to use https if device is capable – client-server solution.*
 - *Attach license to JAR/JAD*

Example – Implementing DRM within CAM (cont.)

- CAM used at different times
 - *Preparation during deployment (adding of product)*
 - *Packaging and re-writing of descriptor files during provisioning*

Design features

- J2EE-based implementation in 100% Java
- Conforms to Ellipsus' extensible architecture and accepts third party plug-ins for content analysis
- Can be connected to anti-virus service or other external services.