APPENDIX B

データ オブジェクトの定義

次に、LicenseManager クラスで使用されるデータ オブジェクトの定義を示します。クラス メンバ データの値を取得するには、getter メソッドを使用します。クラス メンバ データの新しい値を設定するには、setter メソッドを使用します。この項では、各クラスの getter メソッドと setter メソッドを示します。プライベート メンバ データについては記載していません。

```
// Data structure for UserToken information.
public class UserToken implement Serializable {
     private String token;// token generated by the server during login.
     private String username; //username in the server account.
    private int m_timeout = 0; //timeout for the token, 0 represents permanent
    private boolean m_ssl_enabled = false;
}

// Data structure for Group information.
public class Group implement Serializable {
    private String m_group;//group name
    private HashSet<String> m_group_access_list; //group access list
    private HashSet<String> m_devices; //devices belong to this group
}

// Data structure for PAK information.
public class PAK implement Serializable {
private String m_internal_id; //internal encrypted PAK id, use as db key
    private String m_pak_name; //PAK name
    private SKU[] m_sku_list; // list of SKU
    private Date m_last_download; //last time download occurred
    private int m_download_status;//status for the download
    private String m_download_by;//username who perform the download
    private String m_pak_type; //PAK type

    private String m_owner; //username who created this PAK
    private String m_display_name; //PAK name to be displayed
    private HashSet<String> m_pak_access_list; //access list for the PAK

}

// Data structure for SKU information.
public class SKU implement Serializable {
    private String m_device_platforms;
    private String m_type; //SKU type, such as counted, uncounted
    private String m_name; //name of the SKU
    private String m_desc; //description for this SKU
    private int m_ordered_qty; //quantity of the SKU ordered
    private int m_available_qty; //quantify of the SKU left
    private String[] m_features; // feature name of the SKU
```

```
    private String m_entitlement; //PAK internal id
    private String[] m_licenseid_list; //list of licenseline id
    private String m_count;//count  available in the ordered SKU
}

// Data structure for the obtained license.
public class License implement Serializable {
     public enum ObtainSource { CISCO, PREINSTALL, UNKNOWN };
     public enum DeployStatus { DEPLOYED, UNDEPLOYED, PARTIALLY_DEPLOYED};

      private String m_license_id; //license id as db key, this is generated by CLM server
      private String[] m_feature_names;  //feature names contain in the
    private String[] m_feature_versions;//feature version
    private String m_file_name; //filename that contains this License,
not used by CLM
    private String m_device_id;//device UDI this license associated with.
    private Date m_obtain_date;//license obtain date
    private ObtainSource m_obtain_source; //where this license obtained from
    private String m_obtain_by; //username who obtained the license
    private DeployStatus m_deploy_status; //license deploy status
     private String[] m_lic_line_ids; //licenseline ids contain in this license
     private String m_pak_id; // PAK id where this license used to obtained from Cisco.com
}

// Data structure for the obtained licenseline.
public class LicenseLine implement Serializable {
    public enum LLDeployStatus { DEPLOYED, UNDEPLOYED };
    public enum LLType { EVALUATION, EXTENSION, PERMANENT, UNKNOWN};
    public enum LLExpireType { DAY_BASE, DATE_BASE };
    public enum LLState { ACTIVE_IN_USE, ACTIVATE_NOT_IN_USE, INACTIVATE, UNKNOWN }

    private String m_license_line_id; //licenseline ID to identify this licenseline, used
as db key
    private String m_feature_name;//feature name of this licenseline
    private String m_license_xml; //xml string for this licenseline
    private String m_udi;//device UDI this licenseline associated with
    private String m_license_id;//license ID of this licenseline associated with
    private String m_pak_id; //PAK ID of this licenseline associated with
    private LLDeployStatus m_deploy_status = LLDeployStatus.UNDEPLOYED;//Licenseline
deploy status

    private String m_annotation; //Annotation string for this licenseline
    private String m_sku_name; //SKU name this licenseline associated with
    private String m_deployed_by_user; //username who deployed this licenseline
    private Date m_deployed_date; //The date of this licenseline be deployed
    private LLState m_state; //The state of this licenseline read from the device.
    private boolean m_eula_required; //Is EULA required from this licenseline?
    private LLType m_type;//Licenseline type, check LLType enum for options.
    private ExpiryInfo m_expiry_info; //Expiry information if this licenseline is an
expiry license
    private UsageInfo m_usage_info; //Usage left for this licenseline if this licenseline
is an expiry license.

    private long m_index; //Storage index in the device for this licenseline
    private String m_storage;//Storage name of this licenseline is kept
    private CountInfo m_count_info;//CountInfo that keep how many count in the licenseline
//if this is a counted licenseline

    private String m_version; //licenseline feature version
}

Public ExpiryInfo implements Serializable{
    private LLExpireType expire_type; // DAY_BASE, DATE_BASE
    private int expire_days; //number of days to expire for DAY_BASE
```

```
    private int expire_days_from_last_poll; //number of days to expire from last poll, for
    DAY_BASE
    private Date start day; //start date for DATE_BASE
     private Date end_day; //end date for DATE_BASE
}

public class UsageInfo {
    public long m_totalUsage; //Total usage in milli-seconds read from the device
    public long m_usageLeft; //Usage left in milli-seconds read from the device
}

public class CountInfo {
    public int m_totalCount; //Total count of the licenseline read from the device
    public int m_usageCount; //Count used of the licenseline read from the device
}

// Data structure for RehostInfo
Public class RehostInfo implement Serializable{
    private String permission_ticket; //Permission Ticket returned from Cisco.com
    private String rehost_ticket; //Rehost Ticket after Using the Permission Ticket to
revoke a licenseline
    private SKU[] sku_selection; //SKU selected for Rehost
    private String src_device_id; //source device UDI
    private String dest_device_id;//detination device UDI
}

// Data structure for new licences and devices

public class DiscoverySetting implements Serializable
{
       private TransportMethod[] m_trans_methods;
       private String m_group=null;
       private DiscoveryAuthInfo m_discovery_auth;
       private String m_net_ip_addr = null;
       private String m_net_mask = null;
       private String[] m_policy_ids=null;

       public DiscoverySetting(String net_ip, String net_mask, TransportMethod[] methods,
String group, DiscoveryAuthInfo auth_info, String[] policy_ids);
       public void setTransportMethods(Device.TransportMethod[] methods);
       public void setGroup(String group);
       public void setDiscoveryAuthInfo(DiscoveryAuthInfo auth);
       public void setNetIpAddr (String net_ip);
       public void setNetMask(String mask);

       public void setPolicyIds(String[] ids);
       public TransportMethod[] getTransportMethods();
       public String getGroup();
       public DiscoveryAuthInfo getDiscoveryAuthInfo();
       public String getNetIpAddr();
       public String getNetMask();
        public String[] getPolicyIds();
}


public class DiscoverySchedule extends Schedule implements Serializable
{
    private DiscoverySetting[] m_discovery_settings;
    private String[] m_policy_ids;

    public DiscoverySchedule();
    public DiscoverySchedule(DiscoverySetting[] infos, String[] policy_ids);
    public DiscoverySetting[] getDiscoverySettings();
    public void setDiscoverySettings(DiscoverySetting[] infos);
```

```
    public String[] getPolicyIds();
    public void setPolicyIds(String[] ids);
}


public class ExpiringLicenseStatus extends Status implements Serializable
{
    private ExpiringLicenseStatusItem[] m_item_list;
    public ExpiringLicenseStatus();
    public ExpiringLicenseStatusItem[] getExpiringLicenseStatusItems();
    public void setExpiringLicenseStatusItems(ExpiringLicenseStatusItem[] item_list) ;
}


public class ExpiringLicense implements Serializable
{
    private String m_device_id;
    private String m_device_name;
    private String m_feature_name = null;
    private String m_feature_version;
    private String[] m_licline_ids = null;
    private HashMap<String, String> m_feature_map = null;
    private long m_exp;


    public ExpiringLicense(String devid, String devname, String feature, String version,
long exp, String[] licline_ids);
    public ExpiringLicense(String devid, String devname, HashMap<String, String>
feature_map, long exp, String[] licline_ids);

    public String[] getLiclineIDs();
    public String getDeviceID() ;
    public String getDeviceName();
    public String getFeatureName();

    public String getFeatureVersion();
    public HashMap<String, String> getFeatureMap();
    public long getExpiry();
}

public class DevicePlatformInfo implements Serializable
{
    public enum LicenseStorageType {SAFENET, FLEXLM, PIX, ASA, CSCSSM, URLF, UNKNOWN};
    private Device.TransportMethod m_trans_method;
    private int m_port;
    private String[] m_device_access_info = null;
    private DeviceAuthentication m_device_auth;
    private Device.DevicePlatform m_device_platform;
    private Device.DeviceState m_state=Device.DeviceState.DEV_CANNOT_CONNECT;
    private Device.LicenseOperation[] m_device_lic_capabilities = null;

    private String[] m_licensable_features = null;
    private FeatureInfo[] m_feature_info = null;
    private PIXASAFeatureInfo m_pixasa_feature_info = null;
    private SANOSDCOSFeatureInfo[] m_sanosdcos_feature_info = null;
    private LicenseStorageType m_license_storage = LicenseStorageType.UNKNOWN;
    private String m_domain = null;
    private int iosxr_operation_id;

    public DevicePlatformInfo(Device.TransportMethod method, int port);
    public void setTransportMethod(Device.TransportMethod method);
    public void setTransportPort(int port);
    public void setDeviceAccessInfo(String[] info);
    public void setDeviceAuthentication (DeviceAuthentication auth);
```

```
    public void setDevicePlatform(Device.DevicePlatform platform);
    public void setConnectState(Device.DeviceState state);
    public void setDeviceCapabilities(Device.LicenseOperation[] capability);
    public void setLicensableFeatures(String[] features);
    public void setFeatureInfo (FeatureInfo[] feature_info);
    public Device.TransportMethod getTransportMethod();
    public int getTransportPort();

   public String[] getDeviceAccessInfo();
   public DeviceAuthentication getDeviceAuthentication ();
   public Device.DevicePlatform getDevicePlatform();
   public Device.DeviceState getConnectState();
   public Device.LicenseOperation[] getDeviceLicCapabilities();
   public String[] getLicensableFeatures();
   public FeatureInfo[] getFeatureInfo ();
   public LicenseStorageType getLicenseStorageType();

    public void setLicenseStorageType(LicenseStorageType license_storage_type);
    public PIXASAFeatureInfo getPIXASAFeatureInfo();
    public void setPIXASAFeatureInfo(PIXASAFeatureInfo m_pixasa_feature_info);
    public SANOSDCOSFeatureInfo[] getSANOSDCOSFeatureInfo();
    public void setSANOSDCOSFeatureInfo(SANOSDCOSFeatureInfo[] m_sanosdcos_feature_info);

    public int getIOSXROperationID();

    public void setIOSXROperationID(int iosxr_operation_id);

    public String getDomain();

    public void setDomain(String m_domain);
}


// Data structure for devices.
public class Device implement Serializable {
    public enum PollStatus { FAILURE, SUCCESS };
    public enum DeviceState { DEV_CONNECT, DEV_CANNOT_PING, DEV_CANNOT_AUTHENTICATE,
DEV_CANNOT_CONNECT }
    public enum TransportMethod {HTTP, HTTPS, TELNET, SSH};
    public enum LicenseOperation {INSTALL, CLEAR, ANNOTATE, REVOKE};
    private String m_pid; //device product ID
    private String m_vid; //device version ID
    private String m_sn; // device serial number
 private String m_device_id; // device ID, same as m_udi
    private String m_ipaddr; // device IP address
    private String m_udi; // device UDI, a string contains m_pid, m_vid and m_sn, used as
key in db
    private String m_hostname; //hostname of the device
    private String m_display_name; //display name for the device, usually same as hostname
or UDI,
if not given by user
    private String[] m_device_access_info; // device access info list, first string is the
TransportMethod,
                  //second string is device HTTP url if HTTP/HTTPS is used
                  //third string is license agent version.
    private String[] m_licensable_features; //license feature names that are supported by
the IOS image of the device
    private Date m_last_polling_date; //last performed  poll date
    private PollStatus m_last_polling_status; //last polled status
    private String m_polled_by; //username who initiated the poll
    private String[] m_member_device_ids; //device UDI list of the child devices,
applicable to master device in
// stackable devices.
```

```
    private String m_parent_device_id; //device UDI of the parent device, applicable to
the child devices
//in stackable devices.
    private String m_credential; //device credential, internal generated string to assure
device credibility.
    private DeviceAuthentication m_device_auth; //device login information such as
username/password
    private String m_device_model; //device model
    private String m_device_type; //device type
    private HashSet<String> m_device_access_list; //device access list to control user
access.
    private DeviceState m_state; //device connectivity state, check values of DeviceState
    private LicenseOperation[] m_device_lic_capabilities;//device capability toward
licenses,
//such as support resend, rehost
    private String m_sw_version; //software version number for this device
    private FeatureInfo[] m_feature_info;  //a data structure that organizes licenselines
by feature name
    private String[] m_feature_versions; //feature version list mapping to
m_licensable_feature list.
    private boolean m_rma; //boolean value to indicate if this device is RMA'd
    private boolean m_dirty;//boolean value to indicate if this device is dirty and
requires polling.
      private int m_slot; //slot number of this device,
//applicable to stackable devices.
}

// Data structure for UserInfo.
public class UserInfo implements Serializable {
    public enum Role { ADMIN, INVENTORYMGR, PAKMGR, LICENSEMGR, REPORTMGR}
    private String m_username;
    private String m_password;
    private String m_first_name;
    private String m_last_name;
    private String m_company_name;
    private String m_email_address;
    protected boolean m_email_enabled; //enable email notification
    private String m_cco_username; //Cisco.com username
    private String m_cco_password;//Cisco.com password
    private Role m_role; //Associated Role of this user
    private boolean m_has_agree_eula; //boolean value to indicate this user has agree to
EULA
    private boolean m_do_not_ask_eula_again; //boolean value to indicate the user do not
want to be querried again
                                  //once the EULA is agreed.
    private Date m_eula_accepted_time; //Date the EULA is accepted by the user
    private boolean m_show_exp_lic; //boolean value to indicate when user
// login, GUI will show expired licenses.
}

// Data structure to be used when requesting for a license.
public class LicenseRequest {
private SKU[] sku_selection;
private String device_id;
private int request_status;
}


// Data structure to be used when requesting license re-host.
public class RehostRequest {
         private SKU[] sku_selection;
         private String src_device_id;
private String dest_device_id;
}
```

```
// Data structure for device username and password.
public class DeviceAuthentication implements Serializable {
     private String username;
private String password;
private String enable_password;
}

Public class UserPwd {
    public String username;
public String password;
}

public class DiscoveryAuthInfo {
private UserPwd[] auth_pairs;
private String[] enable_password;
}

// Data structure for FeatureInfo
Public class FeatureInfo implements Serializable{
    private LinkedList<LicenseLineInfo> ll_info_list; //list of licenselineInfo
    private long total_expiry_usage;/./total usage left for this license feature, the
value is in milli-seconds
    private String feature_name ; //feature name
    private String feature_version;//feature version
}

public class LicenseLineInfo implements Serializable {
    public String m_licline_id; //licenseline ID
    public UsageInfo m_usage_info; //exipring usage information for each licenseline
    public LicenseLine.LLType m_type; //licenseline type
    public CountInfo m_count_info;//structure contains count of licenseline if the
licenseline is of counted license.
    public LicenseLine.LLDeployStatus m_deploy_status;//licenseline deploy status
    public LicenseLine.LLState m_state;//licenseline deploy state
}

// Data structure for returning ID and status
public class Status implements Serializable {
    private int m_error_code;
    private String m_error_message;
}
public class IDStatus extends Status implements Serializable {
        private IDStatusItem[] m_item_list;
}
public class IDStatusItem extends Status implements Serializable {
    private String m_id;
}

// Data structure for returning PAK and status
public class Status implements Serializable {
    private int m_error_code;
    private String m_error_message;
}

public class PAKStatus extends Status implements Serializable {
    private PAKStatusItem[] m_item_list;
}

public class PAKStatusItem extends Status implements Serializable {
    private PAK m_pak;
}
// Data structure for returning License and status
public class Status implements Serializable {
```

```
    private int m_error_code;
    private String m_error_message;
}
public class LicenseStatus extends Status implements Serializable {
    private LicenseStatusItem[] m_item_list;
}
public class LicenseStatusItem extends Status implements Serializable {
    private License m_license;
}

public class LicLineStatus extends Status implements Serializable {
    private LicLineStatusItem[] m_item_list;
}
public class LicLineStatusItem extends Status implements Serializable {
    private LicenseLine m_licline;
}

public class GroupStatus extends Status implements Serializable {
    private GroupStatusItem[] m_item_list;
}
public class GroupStatusItem extends Status implements Serializable {
    private Group m_group;
}

public class RehostInfoStatus extends Status implements Serializable {
    private RehostInfoStatusItem[] m_item_list;
}
public class RehostInfoStatusItem extends Status implements Serializable {
    private RehostInfo m_group;
}

public class EulaStatus extends Status implements Serializable {
    private boolean accepted_eula;
         private boolean do_not_ask_again;
}

public class ClmJobStatusItem extends Status implements Serializable{
    private ClmJob m_job;
}

public class ClmJobStatus extends Status implements Serializable {
    private ClmJobStatusItem[] m_item_list;
}

// Data structure for returning Device and status
public class Status implements Serializable {
    private int m_error_code;
    private String m_error_message;
}

public class DeviceStatus extends Status implements Serializable {
    private DeviceStatusItem[] m_item_list;
}

public class DeviceStatusItem extends Status implements Serializable {
    private Device m_device;
}


// Data structure for returning ID and status
public class IDStatus extends Status implements Serializable {
    private int m_error_code;
    private String m_error_message;
    private IDStatusItem[] m_item_list;
```

```
}
public class IDStatusItem extends Status implements Serializable {
    private int m_error_code;
    private String m_error_message;
    private String m_id;
}

// Data structure for returning asynchronous Progress
public class ProgressStatus extends Status implements Serializable {
private int m_error_code;
private String m_error_message;
    private String[] current_process_ids;
    private int total_task;
    private int num_processed;
    private int overall_percentage;
    private String[] function_history;
private int num_dev_found;
private Date exec_time;
}

// Interface for asynchronous request status listener.
public interface IDStatusListener {
    public void onStatus(IDStatus status);
}


// Data structure for notifications.
public class Notification {
publice enum Operation={LIC_INSTALL, LIC_CLEAR, LIC_REVOKE, LIC_ANNOTATE,
    DEV_CONNECT, DEV_CANNOT_PING, DEV_CANNOT_AUTHENTICATE,
    DEV_CANNOT_CONNECT, DEV_RMA, DEV_RMA_BUT_RUNNING }
    public enum DevModel={TBD};
    public enum DevType={UNKNOWN};
    public enum Severity={SEVERE, MODERATE, LOW, INFO};

private String message; //message of what this notification is about
private Date time_stamp;//Date of the notification sent
private String dev_id; //device ID of the involved device
private String dev_name; //device name of the involved device
private DevType dev_type; //device type
private DevModel dev_model; //device model
private Severity severity; //severity level, using X.733 standard
private string[]  group; //group list the device associated with
private Operation op; //enum value to indicate defined operation
private string licline_id; //involved licenseline ID
private string feature_name; //involved featuren name
}

// Interface for notification listener.
public interface NotificationListener {
    public void onNotification(Notification notification);
}

public class AuditTrail implements Serializable {
    public enum OperationStatus { SUCCESS, FAILURE }

    private String m_id;
    private String m_user;
    private Date m_start_time;
    private Date m_end_time;
    private String m_operation;
    private OperationStatus m_operation_status;
    private String m_msg;
```

```
    private String m_affected; //involved elements, usually a list of device IDs,
licenseline IDs or PAK IDs.
}

// Data structure for EulaInfo statement and agreement
public class EulaInfo {
    private const String eula_agreement_statement="
End User License Agreement
IMPORTANT: PLEASE READ THIS END USER LICENSE AGREEMENT
CAREFULLY. DOWNLOADING, INSTALLING OR USING CISCO OR CISCO-
SUPPLIED SOFTWARE CONSTITUTES ACCEPTANCE OF THIS
AGREEMENT.

CISCO IS WILLING TO LICENSE THE SOFTWARE TO YOU ONLY UPON
THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN
THIS LICENSE AGREEMENT.  BY DOWNLOADING OR INSTALLING THE
SOFTWARE, OR USING THE EQUIPMENT THAT CONTAINS THIS
SOFTWARE, YOU ARE BINDING YOURSELF AND THE BUSINESS ENTITY
THAT YOU REPRESENT (COLLECTIVELY, "CUSTOMER") TO THIS
AGREEMENT.  IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS
AGREEMENT, THEN CISCO IS UNWILLING TO LICENSE THE SOFTWARE
TO YOU AND (A) DO NOT DOWNLOAD, INSTALL OR USE THE SOFTWARE,
AND (B) YOU MAY RETURN THE SOFTWARE FOR A FULL REFUND, OR, IF
THE SOFTWARE IS SUPPLIED AS PART OF ANOTHER PRODUCT, YOU
MAY RETURN THE ENTIRE PRODUCT FOR A FULL REFUND.  YOUR RIGHT
TO RETURN AND REFUND EXPIRES 30 DAYS AFTER PURCHASE FROM
CISCO OR AN AUTHORIZED CISCO RESELLER, AND APPLIES ONLY IF
YOU ARE THE ORIGINAL END USER PURCHASER.

The following terms of this End User License Agreement ("Agreement")
govern Customer's access and use of the Software, except to the extent  (a)
there is a separate signed agreement between Customer and Cisco
governing Customer's use of the Software or (b) the Software includes a
separate "click-accept" license agreement as part of the installation and/or
download process. To the extent of a conflict between the provisions of the
foregoing documents, the order of precedence shall be (1) the signed
agreement, (2) the click-accept agreement, and (3) this End User License
Agreement.

License.  Conditioned upon compliance with the terms and conditions of this
Agreement, Cisco Systems, Inc. or its subsidiary licensing the Software instead
of Cisco Systems, Inc. ("Cisco"), grants to Customer a nonexclusive and
nontransferable license to use for Customer's internal business purposes the
Software and the Documentation for which Customer has paid the required
license fees. "Documentation" means written information (whether contained in
user or technical manuals, training materials, specifications or otherwise)
specifically pertaining to the Software and made available by Cisco with the
Software in any manner (including on CD-Rom, or on-line).

Customer's license to use the Software shall be limited to, and Customer shall
not use the Software in excess of, a single hardware chassis or card or such
number and types of agent(s), concurrent users, sessions, IP addresses, port(s),
seat(s), server(s), site(s), features and feature sets as are set forth in the
applicable Purchase Order which has been accepted by Cisco and for which
Customer has paid to Cisco the required license fee.

Unless otherwise expressly provided in the Documentation, Customer shall use
the Software solely as embedded in, for execution on, or (where the applicable
documentation permits installation on non-Cisco equipment) for communication
with Cisco equipment owned or leased by Customer and used for Customer's
internal business purposes.  NOTE:  For evaluation or beta copies for which
Cisco does not charge a license fee, the above requirement to pay license fees
does not apply.
```

General Limitations. This is a license, not a transfer of title, to the Software and Documentation, and Cisco retains ownership of all copies of the Software and Documentation. Customer acknowledges that the Software and Documentation contain trade secrets of Cisco, its suppliers or licensors, including but not limited to the specific internal design and structure of individual programs and associated interface information. Accordingly, except as otherwise expressly provided under this Agreement, Customer shall have no right, and Customer specifically agrees not to:

(i)　transfer, assign or sublicense its license rights to any other person or entity, or use the Software on unauthorized or secondhand Cisco equipment, and Customer acknowledges that any attempted transfer, assignment, sublicense or use shall be void;

(ii)　make error corrections to or otherwise modify or adapt the Software or create derivative works based upon the Software, or permit third parties to do the same;

(iii)　reverse engineer or decompile, decrypt, disassemble or otherwise reduce the Software to human-readable form, except to the extent otherwise expressly permitted under applicable law notwithstanding this restriction;

(iv)　use or permit the Software to be used to perform services for third parties, whether on a service bureau or time sharing basis or otherwise, without the express written authorization of Cisco; or

(v)　disclose, provide, or otherwise make available trade secrets contained within the Software and Documentation in any form to any third party without the prior written consent of Cisco. Customer shall implement reasonable security measures to protect such trade secrets.

To the extent required by law, and at Customer's written request, Cisco shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of Cisco's applicable fee, if any. Customer shall observe strict obligations of confidentiality with respect to such information and shall use such information in compliance with any applicable terms and conditions upon which Cisco makes such information available.

Software, Upgrades and Additional Copies. For purposes of this Agreement, "Software" shall include (and the terms and conditions of this Agreement shall apply to) computer programs, including firmware, as provided to Customer by Cisco or an authorized Cisco reseller, and any upgrades, updates, bug fixes or modified versions thereto (collectively, "Upgrades") or backup copies of the Software licensed or provided to Customer by Cisco or an authorized Cisco reseller. NOTWITHSTANDING ANY OTHER PROVISION OF THIS AGREEMENT: (1) CUSTOMER HAS NO LICENSE OR RIGHT TO USE ANY ADDITIONAL COPIES OR UPGRADES UNLESS CUSTOMER, AT THE TIME OF ACQUIRING SUCH COPY OR UPGRADE, ALREADY HOLDS A VALID LICENSE TO THE ORIGINAL SOFTWARE AND HAS PAID THE APPLICABLE FEE FOR THE UPGRADE OR ADDITIONAL COPIES; (2) USE OF UPGRADES IS LIMITED TO CISCO EQUIPMENT FOR WHICH CUSTOMER IS THE ORIGINAL END USER PURCHASER OR LESSEE OR WHO OTHERWISE HOLDS A VALID LICENSE TO USE THE SOFTWARE WHICH IS BEING UPGRADED; AND (3) THE MAKING AND USE OF ADDITIONAL COPIES IS LIMITED TO NECESSARY BACKUP PURPOSES ONLY.

Proprietary Notices. Customer agrees to maintain and reproduce all copyright and other proprietary notices on all copies, in any form, of the Software in the same form and manner that such copyright and other proprietary notices are included on the Software. Except as expressly authorized in this Agreement, Customer shall not make any copies or duplicates of any Software without the prior written permission of Cisco.

Term and Termination. This Agreement and the license granted herein shall remain effective until terminated. Customer may terminate this Agreement and the license at any time by destroying all copies of Software and any Documentation. Customer's rights under this Agreement will terminate immediately without notice from Cisco if Customer fails to comply with any provision of this Agreement. Upon termination, Customer shall destroy all copies of Software and Documentation in its possession or control. All confidentiality obligations of Customer and all limitations of liability and disclaimers and restrictions of warranty shall survive termination of this Agreement. In addition, the provisions of the sections titled "U.S. Government End User Purchasers" and "General Terms Applicable to the Limited Warranty Statement and End User License" shall survive termination of this Agreement.

Customer Records. Customer grants to Cisco and its independent accountants the right to examine Customer's books, records and accounts during Customer's normal business hours to verify compliance with this Agreement. In the event such audit discloses non-compliance with this Agreement, Customer shall promptly pay to Cisco the appropriate license fees, plus the reasonable cost of conducting the audit.

Export. Software and Documentation, including technical data, may be subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. Customer agrees to comply strictly with all such regulations and acknowledges that it has the responsibility to obtain licenses to export, re-export, or import Software and Documentation.

U.S. Government End User Purchasers. The Software and Documentation qualify as "commercial items," as that term is defined at Federal Acquisition Regulation ("FAR") (48 C.F.R.) 2.101, consisting of "commercial computer software" and "commercial computer software documentation" as such terms are used in FAR 12.212. Consistent with FAR 12.212 and DoD FAR Supp. 227.7202-1 through 227.7202-4, and notwithstanding any other FAR or other contractual clause to the contrary in any agreement into which this End User License Agreement may be incorporated, Customer may provide to Government end user or, if this Agreement is direct, Government end user will acquire, the Software and Documentation with only those rights set forth in this End User License Agreement. Use of either the Software or Documentation or both constitutes agreement by the Government that the Software and Documentation are "commercial computer software" and "commercial computer software documentation," and constitutes acceptance of the rights and restrictions herein.
Limited Warranty
Subject to the limitations and conditions set forth herein, Cisco warrants that commencing from the date of shipment to Customer (but in case of resale by an authorized Cisco reseller, commencing not more than ninety (90) days after original shipment by Cisco), and continuing for a period of the longer of (a) ninety (90) days or (b) the warranty period (if any) expressly set forth as applicable specifically to software in the warranty card accompanying the product of which the Software is a part (the "Product") (if any): (a) the media on which the Software is furnished will be free of defects in materials and workmanship under normal use; and (b) the Software substantially conforms to the Documentation. The date of shipment of a Product by Cisco is set forth on the packaging material in which the Product is shipped. Except for the foregoing, the Software is provided AS IS. This limited warranty extends only to the Customer who is the original licensee. Customer's sole and exclusive remedy and the entire liability of Cisco and its suppliers and licensors under this limited warranty will be (i) replacement of defective media and/or (ii) at Cisco's option, repair, replacement, or refund of the purchase price of the Software, in both cases subject to the condition that any error or defect constituting a breach of this limited warranty is reported to Cisco or the party supplying the Software to Customer, if different than Cisco, within the warranty period. Cisco or the party supplying the Software to Customer may, at its option, require return of the Software as a condition to the remedy. In no event does Cisco warrant that the Software is error free or that Customer will be able to operate the Software without problems or interruptions. In addition, due to the continual development

of new techniques for intruding upon and attacking networks, Cisco does not warrant that the Software or any equipment, system or network on which the Software is used will be free of vulnerability to intrusion or attack.
Restrictions. This warranty does not apply if the Software, Product or any other equipment upon which the Software is authorized to be used (a) has been altered, except by Cisco or its authorized representative, (b) has not been installed, operated, repaired, or maintained in accordance with instructions supplied by Cisco, (c) has been subjected to abnormal physical or electrical stress, misuse, negligence, or accident; or (d) is licensed, for beta, evaluation, testing or demonstration purposes for which Cisco does not charge a purchase price or license fee.
DISCLAIMER OF WARRANTY. EXCEPT AS SPECIFIED IN THIS WARRANTY, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OR CONDITION OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, SATISFACTORY QUALITY, NON-INTERFERENCE, ACCURACY OF INFORMATIONAL CONTENT, OR ARISING FROM A COURSE OF DEALING, LAW, USAGE, OR TRADE PRACTICE, ARE HEREBY EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW AND ARE EXPRESSLY DISCLAIMED BY CISCO, ITS SUPPLIERS AND LICENSORS. TO THE EXTENT AN IMPLIED WARRANTY CANNOT BE EXCLUDED, SUCH WARRANTY IS LIMITED IN DURATION TO THE EXPRESS WARRANTY PERIOD. BECAUSE SOME STATES OR JURISDICTIONS DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, THE ABOVE LIMITATION MAY NOT APPLY. THIS WARRANTY GIVES CUSTOMER SPECIFIC LEGAL RIGHTS, AND CUSTOMER MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION. This disclaimer and exclusion shall apply even if the express warranty set forth above fails of its essential purpose.
General Terms Applicable to the Limited Warranty Statement and End User License Agreement

Disclaimer of Liabilities. REGARDLESS WHETHER ANY REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE OR OTHERWISE, IN NO EVENT WILL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY LOST REVENUE, PROFIT, OR LOST OR DAMAGED DATA, BUSINESS INTERRUPTION, LOSS OF CAPITAL, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL, OR PUNITIVE DAMAGES HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY OR WHETHER ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE OR OTHERWISE AND EVEN IF CISCO OR ITS SUPPLIERS OR LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event shall Cisco's or its suppliers' or licensors' liability to Customer, whether in contract, tort (including negligence), breach of warranty, or otherwise, exceed the price paid by Customer for the Software that gave rise to the claim or if the Software is part of another Product, the price paid for such other Product. BECAUSE SOME STATES OR JURISDICTIONS DO NOT ALLOW LIMITATION OR EXCLUSION OF CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.
Customer agrees that the limitations of liability and disclaimers set forth herein will apply regardless of whether Customer has accepted the Software or any other product or service delivered by Cisco. Customer acknowledges and agrees that Cisco has set its prices and entered into this Agreement in reliance upon the disclaimers of warranty and the limitations of liability set forth herein, that the same reflect an allocation of risk between the parties (including the risk that a contract remedy may fail of its essential purpose and cause consequential loss), and that the same form an essential basis of the bargain between the parties.

The validity, interpretation, and performance of this Warranty and End User License shall be controlled by and construed under the laws of the State of California, United States of America, as if performed wholly within the state and without giving effect to the principles of conflict of laws, and the State and federal courts of California shall have jurisdiction over any claim arising under this

Agreement. The parties specifically disclaim the UN Convention on Contracts for the International Sale of Goods. Notwithstanding the foregoing, either party may seek interim injunctive relief in any court of appropriate jurisdiction with respect to any alleged breach of such party's intellectual property or proprietary rights.If any portion hereof is found to be void or unenforceable, the remaining provisions of the Agreement shall remain in full force and effect. Except as expressly provided herein, this Agreement constitutes the entire agreement between the parties with respect to the license of the Software and Documentation and supersedes any conflicting or additional terms contained in any purchase order or elsewhere, all of which terms are excluded. This Agreement has been written in the English language, and the parties agree that the English version will govern.

Supplemental End User License Agreement for Trial Use of Software with Cisco License Manager Software
IMPORTANT-READ CAREFULLY: This Supplemental License Agreement for Trial Use of Software with Cisco License Manager Software ("Supplement") contains additional limitations on the license to the Software provided to Customer under the End User License Agreement between Customer and Cisco. Capitalized terms used in this S upplement and not otherwise defined herein shall have the meanings assigned to them in the End User License Agreement. To the extent that there is a conflict among any of these terms and conditions applicable to the Software, the terms and conditions in th i s Supplement shall take precedence.

By installing, allowing to be installed, downloading, accessing or otherwise using the Software or using the equipment that contains this Software, Customer agrees to be bound by the terms of this Supplement . If Customer does not agree to the terms of this Supplement , Customer may not install, download, access or otherwise use the Software. Customer may retain a third party ("Contractor") to install the Software for Customer, provided that (i) Customer and Contractor agree that the provisioning of installation services by Contractor to Customer creates an agency relationship and (ii) Customer shall remain fully responsible and liable for compliance by Customer and Contractor with the terms of the End User License Agreement and this S u pplement . When used below, the term "server" refers to a central processor unit owned or leased by Customer or otherwise embedded in equipment provided by Cisco.

Section 1. Customer hereby agrees that the terms of this S upplement shall govern use of all Cisco Software provided to Customer solely for trial use (e.g. , a license of 30 or 60 days) on any devices in your network that use the Software in conjunction with the license of Cisco License Manager , unless otherwise specifically agreed by Customer and Cisco. In the event of a conflict in terms between this S upplement and the terms of the End User License Agreement the terms of this S upplement shall apply.

Section 2. DISCLAIMER OF WARRANTY. ALL SOFTWARE LICENSED UNDER THIS SUPPLEMENT IS PROVIDED "AS IS". ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OR CONDITION OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, SATISFACTORY QUALITY, NON-INTERFERENCE, ACCURACY OF INFORMATIONAL CONTENT, OR ARISING FROM A COURSE OF DEALING, LAW, USAGE, OR TRADE PRACTICE, ARE HEREBY EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW AND ARE EXPRESSLY DISCLAIMED BY CISCO, ITS SUPPLIERS AND LICENSORS. TO THE EXTENT AN IMPLIED WARRANTY CANNOT BE EXCLUDED, SUCH WARRANTY IS LIMITED IN DURATION TO THE EXPRESS WARRANTY PERIOD. BECAUSE SOME STATES OR JURISDICTIONS DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, THE ABOVE LIMITATION MAY NOT APPLY. THIS WARRANTY GIVES CUSTOMER SPECIFIC LEGAL RIGHTS, AND

```
CUSTOMER MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM
JURISDICTION TO JURISDICTION.  This disclaimer and exclusion shall apply even if
the express warranty set forth above fails of its essential purpose.


Section 3.   You hereby agree that  the trial  use of the Software provided for any
device in your network under this Supplement is terminable at will by
Cisco.    T he specific license term  for any Software licensed under this
Supplement  and any additional terms covering use of the Software on specific
devices  may be set forth separate documentation by Cisco.  The Software
provided under this Supplement may be shut down or terminated by Cisco after
expiration of the term of each specific license. Cisco reserves the right to
terminate or shut down any such Software electronically or by any other means
available. While alerts or such messages may be provided, it is your sole
responsibility to monitor your  usage of any Software to ensure that your systems
and networks are prepared for the shut down of the Software. You also hereby
agree that Cisco will not have any liability whatsoever for any damages, including,
but not limited to, direct, indirect, special, or consequential damages related to
any Software being shutdown or terminated.



By clicking the "accept" button  or typing "yes" Customer  indicates  that it has
read and agrees to be bound by all the terms provided herein including the terms
of the  End User License Agreement and the Supplement.";
    private boolean I_have_read_and_I_accept_this_EULA;
    private boolean do_not_ask_me_again;
    private Date eula_accepted_time;
}

// Data structure for Schedule
public class Schedule implements Serializable
{
    public enum ClmTask { POLL_LICENSES, CHECK_EXPIRING_LICENSES, DISCOVER_DEVICES };
    private Calendar m_starting_date=Calendar.getInstance();
    private String m_name;
    private Frequency m_frequency=0;
    private int m_notify_if_less_than=1;
    private String m_username;


    private IpMask[] m_ip_masks;

    public Schedule(ClmTask task);

    public String getScheduleName();

    public int getFrequency()

    public int getNotifyIfLessThan()

    public ClmTask getScheduleTask()

    public Calendar getStartingDate()

    public String getUserName()

    public void setFrequency(int frequency)

    public void setNotifyIfLessThan(int i)

    public void setStartingDate(Calendar start)
```

```
    public void setUserName(UserToken token)
}

public class Policy {
    public enum DeviceAttribute {
        MODEL, GROUP
    };
    public enum SKUAttribute {
        FEATURE_NAME
    };
    String id;                 // ID of the policy
    String name;               // name of the policy
    String owner;              // owner of the policy
    String desc;               // description of the policy
    boolean is_public;         // is this policy public?
    DeviceFilter dev_filter;   // the device filer
    SKUFilter sku_filter;      // the SKU filter
    SKUIdentifier sku_in_use;  // the SKU in use by the policy
    String executed_by;        // user who executes the policy
    Date executed_at;        // time when executing the policy
}


public class SKUFilter {
    String feature_name;       // specify the interested feature name
}


public class DeviceFilter {
    String type;               // specify the interested type of device (for future use)
    String model;              // specify the interested model of device
    String sw_version;         // specify the interested dev sw version (for future use)
    String group;              // specify the interested device group
    String ip_lower_limit;     // specify the upper limit of IP addr in a range
    String ip_upper_limit;     // specify the lower limit of IP addr in a range
}

public class SKUIdentifier {
    String pak;                // PAK of the selected SKU
    String sku_name;           // name the selected SKU
}



//report structure
enum OutputFormat={ HTML, TEXT };
enum ReportSubject={LICENSE_DISCREPANCY, REDEPLOYABLE_LICENSE,
           UNUSED_LICENSE, DEVICE_SUMMARY, LICENSE_EXPIRY,
           RMA_DISCREPANCY, AUDITTRAIL};

//ClmJob structure
public class ClmJob implements Serializable{
    private String m_job_id; // Job ID to identify a Job in DB, it is the same as request
ID
    private Date m_start_date; //date when the Job started.
    private Date m_end_date;//date when the Job ends
    private String m_username;//username who initiates the job
    private String m_func_name;//function name of the job is performed
    private boolean m_safe_to_kill=true;//boolean value to indicate if it is safe to
cancel this task
    private String m_comment; //human readable statement for this job
    private ProgressStatus m_progress;//ProgressStatus for this job
    private boolean m_job_done=false;//boolean value to indicate the job has completed.
```

```
    private ModuleLifeCycleIf m_module_handler; //an interface to handle cancel job
instruction
    private IDStatusm_idstatus; //the IDStatus of the operation after it is completed.
}
```