DATALOGIC™

APPLICATION NOTE

Datalogic WebSentinel™ DB Architecture

Reference SW Packages:

Datalogic WebSentinel™: sw package 4.0.0 and later

History

Issue	Date	Change
Rev. 0	August 23rd, 2010	First release

CONTENTS

1 OVERALL ARCHITECTURE

WebSentinel is based on a 3 layers architecture:

1. **Client Layer**: implemented by any Web Browser. There may be several instances of Client Layer, each allocated on its own machine. An instance is initiated when a Web Browser logs on the Server Layer and terminates when the Web Browser is logged off by the Server Layer,

2. **Server Layer**: implemented by a Web Server based application. It interfaces the Client Layer providing it with all windows (pages) and their content. It supports both static and dynamic contents, and the automatic update of windows content. There is a single instance of Server Layer.

3. **Backoffice Layer**: it collects all information from the **Plant**, and makes it available to the Server Layer for display. It also performs all computations and registrations that are necessary to record statistics info based on session and hours periods. There is a single instance of Backoffice Layer.

Server and Backoffice Layers reside on a same machine, but they are executed by 2 different Java virtual machines.
The pair Server and Backoffice Layers will be referred to from now on as WebSentinel.

Client Layers normally reside on different machines, although it is possible for a Client Layer to be activated on the same machine as WebSentinel.

Clients and Server may be part of a same intranet, but it is also possible that they belong to different domains of the internet.

The Plant is made up by several reading stations (laser and vision systems) each interfacing WebSentinel via a single controller.

The connection between WebSentinel and the reading stations is via TCP/IP.
The overall system architecture is depicted in Figure 2.

1.1 MANAGED READING SYSTEM TYPES

Past versions of WebSentinel had to face conflicting requirements:

- On one hand, WebSentinel was expected to be able to interface different types of reading systems, without any specific type related behavior.
- On the other hand WebSentinel was expected to be able to adapt its windows to the characteristics of the reading system it is currently displaying.

WebSentinel allows a better handling of these problem by introducing an explicit management of supported array types.

Based on array type info:

- The maximum number of slave nodes in the array will be automatically configured.
- The caption of slave windows and the prefix of automatically assigned slave names (“scanner” vs. “slave”) will be automatically configured.

- The possible types of a slave of an array are known and can be checked.
- The maximum number and caption of alarm LEDS (both of the controller and of individual slaves) will be automatically configured.
- The number and default label of digital inputs (and the caption of the digital input display) will be automatically configured (N.B. digital inputs may be currently used to display additional alarm info).
- The operating mode of the array may be restricted.
- The way the array is configured (via Genius MIB or not) is known.
- Whether also slave nodes support Genius is known.
- Whether array components allow the access to log files, and in what directory they are available (assumed homogeneous for all array elements).
- Whether array components allow the access to image files, and in what directory they are available (assumed homogeneous for all array elements).

The “Managed Reading System Types” description file includes also information about:

- The specific characteristic of supported slave types (including an unknown type).
- The definition of alarms and their characteristics:
  - Acronym (indicated as “probable cause”)
  - Univocal numeric identifier
  - Category
  - Default severity.

The default severity of each individual alarm type can then be redefined for all arrays of the plant (default plant severity assignment). The severity assignment provided by the “Managed Reading System Types” description file will provide the default value for default plant level assignments.

The default severity of each individual alarm type can then be redefined for each array of the plant (array level severity assignment). The severity assignment provided by the default plant severity assignment will provide the default value for default plant level assignments.

In any case, a specific severity assignment will be defined for each array, even if the array assignment is identical to the plant level default assignment.

In any case, the severity that will be considered is the one of the specific array.

Each reading system type will be described through an XML text and the behavior of WebSentinel will be based on this description.

New reading system types can be added by adding the related description to the XML type file.

1.2 ALARM SEVERITY AND LIFECYCLE

WebSentinel is open to the support of alarm severity.
In some older versions of WebSentinel all alarms share a same severity label.
In the long run one may expect the individual arrays to explicitly indicate the severity of the alarms they are raising: but this implies that the WebSentinel protocol gets upgraded.

An intermediate solution can be based on the manager-based assignment of a severity level to alarms: it will be WebSentinel itself (the Backoffice Layer) that performs severity assignment based on a configurable severity assignment profile table.

A default value of the alarms severity assignment profile is defined for the whole plant, but this default value can be overridden for individual arrays.
The definition of the alarms severity assignment profile is based on the assignment of an integer code to each alarm.

Four levels of severity are supported.

WebSentinel will also be capable to support an alarms lifecycle.
An alarm can be in several states:

- Active, if it has been raised but.
- Cleared, if the alarm doesn't hold any longer.

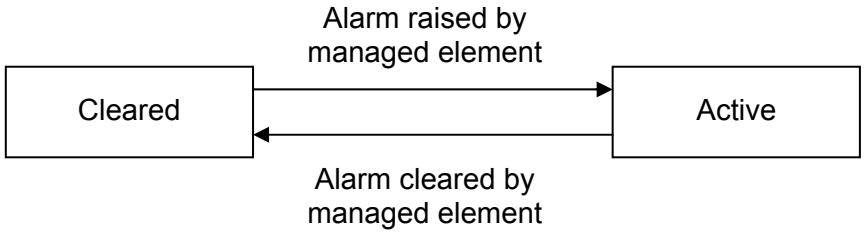

Figure 1 Alarms Lifecycle

Additionally, each alarm cause is assigned a category.

There are also diagnostic events that may not be treated as alarms, since there is no related clear event: the only events of this type currently considered are performance threshold crossing notifications.

Following is a set of constants that supports the definition of the alarms severity assignment profile, category and of an alarms lifecycle.

```
    /**
     * Costanti per la categoria degli allarmi
     */
    static final int communicationAlarmCategory     = 0;//SMTP, TCP e slave
                                                        //comms
    static final int equipmentAlarmCategory         = 1;//motor,laser,...
    static final int enviromentalAlarmCategory      = 2;//digital input
    static final int qualityAlarmCategory           = 3;//no read
    static final int configMismatchAlarmCategory    = 4;//unexpected slaves
    static  final  int  performanceEventCategory          =  5;//performance
threshold
                                                        //crossing
notification
    /**
```

```
 * Costanti per la severita' degli allarmi
 */
static final int clearedSeverity       = 0;
static final int warningSeverity       = 1;
static final int minorSeverity         = 2;
static final int majorSeverity         = 3;
static final int criticalSeverity      = 4;
```

Alarms are referenced through the data item in the WebSentinel PDU that is associated to the alarm.

1.2.1 Definition of Alarms

The set of alarm causes that are supported is defined in the “Managed Reading System Types” description file. For each alarm cause the following info is provided:

1. The acronym of the probable cause.
2. The numeric code of the probable cause.
3. The category.
4. The default severity.

Whilst the first three pieces of information are fixed, the fourth provides only a default value: this default value will be used by the Backoffice Layer to initialize table PLANT_ALMSEVCFG, that defines the default severity of each alarm for the plant (for all arrays of the plant).

Starting from this table, the Backoffice and the Server Layers (in case of recovery from and old .ini file or configuration editing, respectively) will initialize the alarm severity profile for each new array in table DEVALMSEVCFG.

During the on-line behaviour, in order to acquire the correct severity for an alarm cause related to an array, it will be necessary only to access table DEVALMSEVCFG.
For plant level alarms the severity will be taken from the PLANT_ALMSEVCFG table.

1.2.2 Digital Input Signals as Alarms

Digital inputs will be handled as alarm sources:

- Either they are associated to resources (e.g. the encoder), in which case the bit is raised/cleared when an alarm is raised/cleared for the associated resource,
- Or they are connected to an alarm sensor, in which case the bit is raised/cleared when an alarm condition is sensed active/cleared.

In any case digital inputs that are defined as relevant for an array will have to be associated to a defined alarm cause: if no suitable alarm cause is found among those that are defined the generic alarm cause associated to the digital input (GenericInput<k>Alarm) **must** be selected.

Beside an alarm cause a relevant digital input must be associated to a caption (1 to 3 chars), that will be used on the GUI, and will also be used in report files when the digital input is associated to its generic alarm cause.

The overall system architecture is depicted in the following figure:

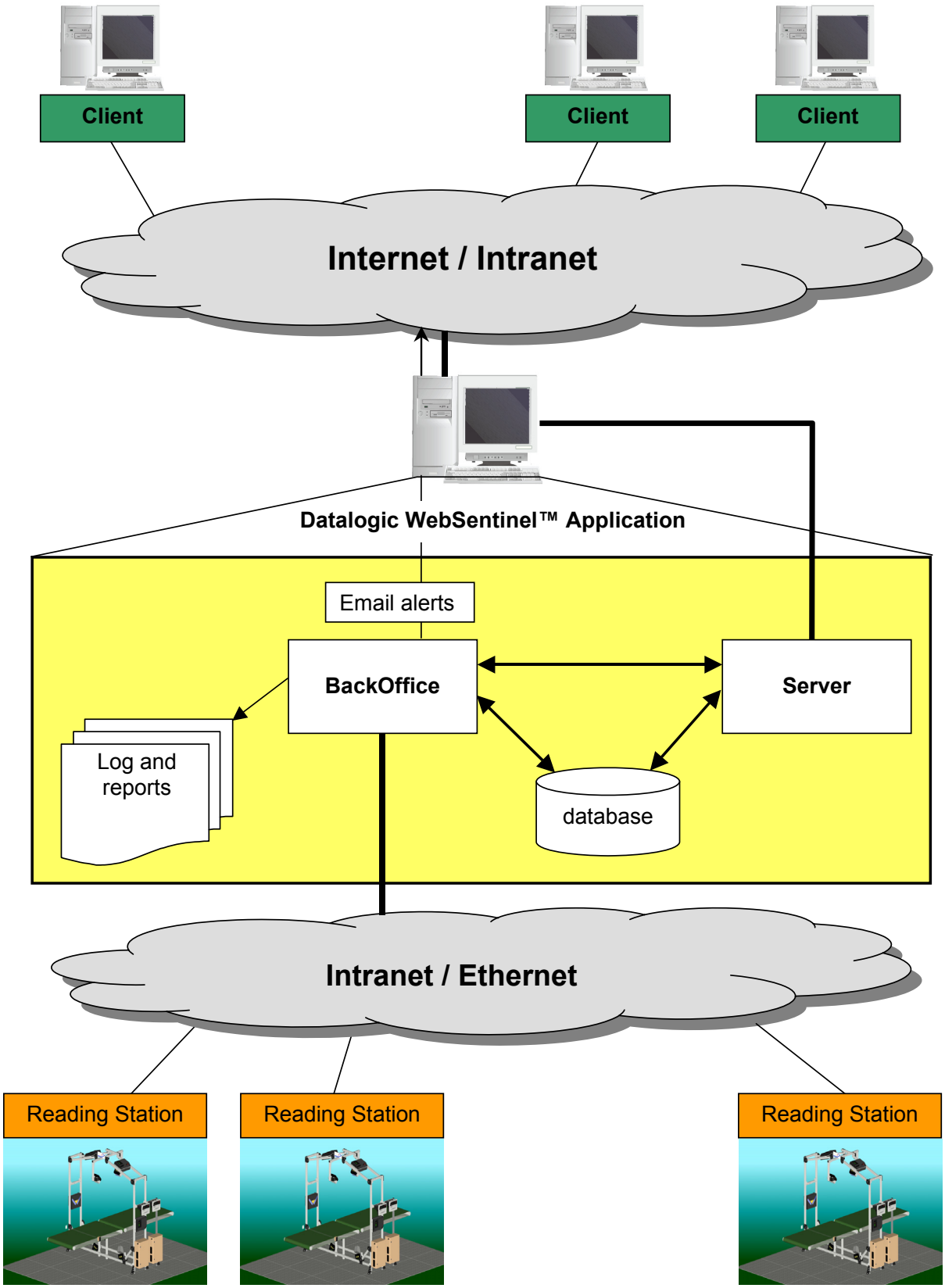

Figure 2 Overall System Architecture

1.3 SERVER-BACKOFFICE INTERFACE

1.3.1 Program Interface vs. Database Interface

The interaction between Server and Backoffice Layers takes place through 2 very different mechanisms:

- Via exchange of data that are contained in the database and that are written by one entity and read by the other.
- Via exchange of synchronization triggers, that allow the two layers to coordinate their operations. These synchronization events are exchanged on a TCP connection (initiator of connection: Server Layer; responder: Backoffice Layer).

1.3.2 Server-Backoffice Program Interface

The following synchronization events will be exchanged between the 2 layers. In particular:

- Events will be implemented via procedure calls.
- Events will be exchanged as RPC inter objects method calls, based on the use of the CAJO support (https://cajo.dev.java.net/).
- All events will be synchronous, with synchronization implemented through the return of the interface procedure.
- Most events will be initiated by the Server Layer, with the Backoffice Layer responding to Server Layer requests.

An additional, implicit event is represented by the fact that the Backoffice Layer accepts the connection request from the Server Layer on termination of its Initializing phase.

1.3.3 Server-Backoffice Database Interface

The database interface is based on the fact that one entity writes a piece of information in the database, and the other entity reads it.

There are 2 types of information:

- Configuration information.

  Normally written by the Server and read by Backoffice.

  The Backoffice Layer writes configuration info only during the Initializing phase or during the static or dynamic plant configuration procedure.

- Alarm and Performance information.

  Written by the Backoffice and read by Server.

The database will be initialized at the first startup of the system by the Backoffice Layer that will also populate it either with the information derived from a pre-existing database file or with default values: in any case all required information will be explicitly present in the database.

The Server Layer will validate as far as possible the configuration info provided by the user:

- It will provide sensible defaults.

- It will provide menus where legal values can be selected from.
- It will perform checks against legal value sets or syntax constraints.

1.3.4 Configuration Procedure

WebSentinel configuration may be modified only when operations are stopped: the Server will stop Backoffice operations using the Server-Backoffice Program Interface.

It will do this only when an explicit configuration change is applied: when this occurs also the operations of Clients will be temporarily interrupted.
The Server Layer issues a Stop command for the Backoffice layer when it does the first configuration change.

The Server Layer issues a Start command for the Backoffice layer when it terminates the reconfiguration procedure: this fact is triggered by the exit from the Settings window after a configuration change has been performed.
By issuing the Start command the Server Layer will trigger the execution of the plant configuration procedure, so that plant configuration will be re-computed by the Backoffice Layer and as a consequence its display will be updated.

1.3.5 Alarms Data

It is expected that in the database they are registered not only in the way they are received from the array, but the Backoffice performs all the semantic mappings (based on array type) and propagations to summary alarm indicators so that the Server is responsible only of display.

The alarm mapping info is part of the description of a reading system type.

1.3.6 Performance Data

Computation of performance indices and generation of history data and reports are the responsibility of Backoffice.

The Server is responsible only of display of performance data.

2 DATABASE ARCHITECTURE

2.1 DATABASE PLATFORM

Derby: Apache Derby, an Apache DB subproject, is a relational database implemented entirely in Java and available under the Apache License (Version 2.0).

The database will be initialized at the first startup of the system by the Backoffice Layer that will also populate it either with the information derived from a pre-existing database file or with default values: in any case all required information will be explicitly present in the database.

A single schema is currently used, named “S”.

Following is the definition of the tables that are part of this schema.

2.2 REMOVAL OF AN ARRAY

When an array is removed from the plant, all rows in all tables related to it must be removed.

When a slave is removed from an array, all rows in all tables related to it must be removed.

2.3 TABLES

2.3.1 DeviceTypes File

Some information that is available in the DeviceTypes2.xml configuration file is replicated inside the database.

This information is described in the following tables:

- SUPDEVSLAVETYPE

SUPDEVSLAVETYPE Table

This table lists the types of all devices (array controller or slave) that may appear in the Plant and are managed by WebSentinel.

Write Access: Backoffice Layer during startup

Read Access: Anyone

Column	Notes	SQL Type
type	String name of the type as listed in file DeviceTypes2.xml	VARCHAR NOT NULL
level	Indicates whether this is the type of an array or that of a slave	INT NOT NULL 1 = array 2 = slave

Index columns: -

2.3.2 Plant Configuration

Plant level configuration is described by 2 tables:

- PLANT_SETUP
- PLANT_ALMSEVCFG

PLANT_SETUP Table

This table provides system (plant) wide information and default values for array and slave parameters. It is a single row table.

Write Access: Backoffice Layer during configuration restore;

Server Layer during system configuration

Read Access: Backoffice Layer: only parameter mulAsGood

Server Layer: display of plantName, usAirportInterfaceStyle to define the look&feel, default values when new arrays are configured

Column	Notes	SQL Type
plantName	Up to 20 UNICODE character string, not null	VARCHAR(20) NOT NULL
packTrack	Default operating mode of the plant's arrays	INT NOT NULL 0 = OnLine 1 = PackTrack
labelInput0	Must be not null if input 0 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause0	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput0 is not NULL
labelInput1	Must be not null if input 1 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
ProbableCause1	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput1 is not NULL
labelInput2	Must be not null if input 2 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
ProbableCause2	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput2 is not NULL

Column	Notes	SQL Type
labelInput3	Must be not null if input 3 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
ProbableCause3	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput3 is not NULL
labelInput4	Must be not null if input 4 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
ProbableCause4	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput4 is not NULL
labelInput5	Must be not null if input 5 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
ProbableCause5	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput5 is not NULL
labelInput6	Must be not null if input 6 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
ProbableCause6	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput6 is not NULL
labelInput7	Must be not null if input 7 is present in default inputs set. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
ProbableCause7	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput7 is not NULL
visibleSlots	Default number of visible slots, for plant's arrays	INT NOT NULL Range = 0..hardwired(40)
slaveLowPerformance	Default low performance threshold for individual slaves: in range 0.0 to 100.00	DOUBLE NOT NULL
lowPerformance	Default low performance threshold for individual arrays: in range 0.0 to 100.00	DOUBLE NOT NULL

Column	Notes	SQL Type
slaveNoReadAlarm	Default length of the sequence of parcels for which no label is read by the slave for which a NoReadAlarm is raised.	INT NOT NULL
noReadAlarm	Default length of the sequence of parcels that are not GoodRead for which a NoReadAlarm is raised.	INT NOT NULL
usAirportInterfaceStyle	Defines whether the slaves of an array are identified as A, B, C, … or whether they are identified as 01, 02, 03, … The configuration field should be renamed as: “slave identification”, with value standard or numeric	INT NOT NULL 0 = standard 1 = numeric
mulAsGood	Specifies whether MultipleReads must be counted by WebSentinel as GoodReads. This parameter applies to all arrays of the plant.	INT NOT NULL 0 = multiple not as good 1 = multiple as good
devicedomain	It represent the default Domain to which machines of the plant belong, if any, otherwise NULL	VARCHAR
deviceusername	It represent the default UserName to be used to remotely access the file system of plant’s machines, otherwise NULL	VARCHAR
devicepassword	It represent the default password to be used to remotely access the file system of plant’s machines, otherwise NULL	VARCHAR

Index columns: -

PLANT_ALMSEVCFG Table

This table provides system (plant) wide default values for the assignment of severity to alarms. This default assignment can be overridden by an array specific assignment.

The table contains one row for each type of alarm that is defined (see paragraph 1.2).

Write Access: Backoffice Layer during configuration restore or at database creation;

Server Layer during system configuration

Read Access: Backoffice Layer: when integrating the alarm profile of a new array;

Server Layer: when it configures the alarm profile of an array (in future only). Currently this table may be ignored.

Column	Notes	SQL Type
probableCause	A non-negative integer, associated one-2-one to an alarm cause	INT NOT NULL
severity	Default severity associated to the related alarm cause	INT NOT NULL 1 = warningSeverity 2 = minorSeverity 3 = majorSeverity 4 = criticalSeverity

Index columns: probableCause

PLANT_SLOTS Table

This table provides system (plant) wide default values for the definition of code slots of each array of the plant. This default assignment can be overridden by an array specific assignment.

The table contains one row for each code slot that is defined in the plant wide default configuration (the max number of rows is hardwired in the code, and it is currently 40).

Write Access: Backoffice Layer during configuration restore;
Server Layer during system configuration

Read Access: Server Layer: default values when new arrays are configured

Column	Notes	SQL Type
slotIndex	Index of the slot	INT NOT NULL Range = 0 .. (PLANT_SETUP.visibleSlots-1)
labelSlot	Default label for this slot, in a plant's array. Up to 10 UNICODE characters.	VARCHAR(10) NOT NULL

Index columns: slotIndex

2.3.3 Arrays Configuration

DEVICECFG Table

Table DEVICECFG provides information about the arrays that are part of the plant.
There will be a row for each array.

Write Access: Backoffice Layer during configuration restore;
Server Layer during system configuration

Read Access: Backoffice Layer: to connect and poll arrays;
Server Layer: to create the TreeView, and to profile the array tabs.

Column	Notes	SQL Type
deviceIndex	A non negative, unique, number. **N.B.: differently from old WebSentinel the deviceIndex of the plant's arrays must not be consecutive.** **This means that arrays may also be deleted.** **In this case all related info is removed from the database.**	INT NOT NULL
name		VARCHAR(20) NOT NULL
type	Must be an element of a predefined set of strings that are the identifiers of the array types defined in the "Managed Reading System Types" description file	VARCHAR(20) NOT NULL
addr	IP address (in decimal dotted notation) of the array controller	VARCHAR(15) NOT NULL
port	TCP port of the WebSentinel agent on the array controller	INT NOT NULL
lowPerformance	Low performance threshold for this array: in range 0.0 to 100.00	DOUBLE NOT NULL
noReadAlarm	Length of the sequence of parcels that are not GoodRead for which a NoReadAlarm is raised for this array	INT NOT NULL
labelInput0	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause0	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput0 is not NULL
labelInput1	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause1	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput1 is not NULL
labelInput2	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)

Column	Notes	SQL Type
probableCause2	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput2 is not NULL
labelInput3	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause3	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput3 is not NULL
labelInput4	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause4	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput4 is not NULL
labelInput5	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause5	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput5 is not NULL
labelInput6	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause6	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput6 is not NULL
labelInput7	Must be not NULL if digital input 0 is present in this array, NULL if it is not used. A null string is illegal. Up to 3 UNICODE characters.	VARCHAR(3)
probableCause7	It represents the probable cause code of the alarm associated to the digital input , if any	INT Not NULL if labelInput7 is not NULL

Column	Notes	SQL Type
packTrack	Operating mode of this array	INT NOT NULL 0 = OnLine 1 = PackTrack 2 = ClusterContinuous
visibleSlots	Number of visible slots for this array	INT NOT NULL Range = 0..hardwired(40)
geniusPortNo	Port Number of the Genius application, in case it is significant and configurable	INT Range=1024..65535 0 if not significant
devicedomain	It represent the Domain to which machines of this array belong, if any, otherwise NULL	VARCHAR
deviceusername	It represent the UserName to be used to remotely access the file system of this array's machines, otherwise NULL	VARCHAR
devicepassword	It represent the password to be used to remotely access the file system of this array's machines, otherwise NULL	VARCHAR
TABINDEX	It is the index of the tab where the icon of the array is displayed in the Plant Layout window	INT DEFAULT NULL

Index columns: deviceIndex

DEVICESLOTSCFG Table

This table provides the definition of code slots of each array of the plant.
The table contains one row for each code slot that is defined for the array (the max number of rows is hardwired in the code, and it is currently 40).

Write Access: Backoffice Layer during configuration restore;

Server Layer during system configuration

Read Access: Server Layer: default values when new arrays are configured

Column	Notes	SQL Type
deviceIndex	Index of the array to which the row is related	INT NOT NULL Range = 0 .. 255
slotIndex	Index of the slot	INT NOT NULL Range = 0 .. (DEVICECFG.visibleSlots-1)
labelSlot	Default label for this slot, in a plant's array. Up to 10 UNICODE characters.	VARCHAR(10) NOT NULL

Index columns: deviceIndex, slotIndex

2.3.4 Slaves Configuration

Table SLAVECFG provides information about the slave nodes that are part of the arrays of the plant.
There will be a row for each slave, be it configured (detected or not) or only detected.

Write Access: Backoffice Layer during configuration restore, or when it detects a slave that was not configured.

Server Layer during configuration. The slave configuration window must include a button that allows to “accept” a slave that has been detected but was not previously configured.

Read Access: Backoffice Layer: to compute NoReadAlarms and performance threshold crossing state.

Server Layer: to create the TreeView, and to profile the scanner tabs.

Column	Notes	SQL Type
deviceIndex	A non negative, unique, number.	INT NOT NULL
slaveIndex	A non negative number, unique inside the array. It is associated to the slave index/address inside the array.	INT NOT NULL
name		VARCHAR(20) NOT NULL
type	“Unknown” in case configured=false or in case of a configuration derived from an existing .ini WebSentinel file.	VARCHAR(20) NOT NULL
lowPerformance	Low performance threshold for this slave: in range 0.0 to 100.00	DOUBLE NOT NULL
noReadAlarm	Length of the sequence of parcels that are not read for which a NoReadAlarm is raised for this slave	INT NOT NULL
configured	Tags whether a slave has been configured by the operator (true) or has only been detected (false). In case the slave has been detected even though it had not been configured its name and type are assigned by the Backoffice Layer	INT NOT NULL 0 = false 1 = true
geniusIPAddr	IP address to be used to connect to the Genius application, in case it is significant and configurable	VARCHAR(15) Decimal dotted notation
geniusPortNo	Port Number of the Genius application, in case it is significant and configurable	INT Range=1024..65535 0 if not significant

Index columns: deviceIndex, slaveIndex

2.3.5 Security Configuration

Table SECURITY_SETUP provides information about users, their passwords and their access rights, and the preferred language.
There will be a row for each user (account).

Notice that there may be several users with the same access rights. When constructing this table starting from the existing .ini file, 3 normal accounts will be defined (user, operator, administrator). If a password is defined, the same string will be used also as username, otherwise an identical username and password will be defined, based on the access level (operator and administrator).

Write Access: Backoffice Layer: at creation (with default users) or restore time.

Server Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
userName	The log-in identifier of the user	VARCHAR(20) NOT NULL
userPassword	The password of the user: it is kept in encrypted form.	VARCHAR(20) NOT NULL
accessRights		INT NOT NULL 0 = limited user 1 = (normal) user 2 = limited operator 3 = (normal) operator 4 = limited administrator 5 = (normal) administrator
geniusPassthrough	This parameter specifies both whether the functionality is enabled for an account, and the access rights of the Genius session that gets activated. Whilst the database encodes this as an integer, the display and the selection on the GUI should be based on the related text	INT NOT NULL 0 = disabled 1 = enabled – user level 2 = enabled – installer level 3 = enabled – programmer level 4 = enabled – reserved level
language	The preferred language for sessions related to this account.	VARCHAR(20) NOT NULL
unit	The preferred measurement system for sessions related to this account	VARCHAR • imperial • metric

Column	Notes	SQL Type
vnc	Defines whether a username is allowed to connect to WebSentinel or to elements of the Plant using VNC. The single Plant elements that support VNC, must anyway explicitly allow it.	INT NOT NULL DEFAULT 0
pswLastTime	Registers the time of the last password change, in order to check whether a password has become stale.	TIMESTAMP
downloadImagesFileAllowed	Defines whether a username is allowed to require the image download form the array	INT NOT NULL DEFAULT 0

Index columns: userName

2.3.6 Code Groups Configuration

Differently from the past releases of WebSentinel, WebSentinel doesn't assume that code groups are global over all plant's arrays.
Thus, it must be possible to assign a specific description to code groups on a per array basis.

It must also be possible to define per single array the number of significant code groups.
The configuration of code groups that is done at plant level represents only the default configuration of individual array, and can be overridden by specific configurations.

Maximum number of code groups per each array: 40.

2.3.7 Code Types Configuration

It is assumed that code types are global over all plant's arrays.
This is not a real constraint since code types in WebSentinel are actually limited to represent symbologies, and these are identified not by position but by their AIM identifier.
So, in practice, this constraint states only that in a plant there may be a maximum of 10 different symbologies.

Maximum number of code types: 20.

2.3.8 Parcel Archive

The parcel archive contains only information about parcels that have been acquired through messages of type EXTENDED_PARCEL; parcels that have been acquired through messages of type PARCEL are not registered in this archive.

The parcel archive info is provided through tables:

- table PARCEL_ARCHIVE_DIR
- table PARCEL_ARCHIVE_SEQ

- table PARCEL_ARCHIVE_EXT
- table PARCEL _ARCHIVE_SLOTEXT

PARCEL_ARCHIVE_DIR Table

This table contains identification and all barcodes information about all parcels that have been processed by any array of the plant.

There is one row for each barcode of each parcel of each array.

This is the entry point for the search and display of a parcel. Several queries are possible:

1. By deviceIndex + parcelId: fetches info about a single parcel of a specific array: returns all rows related to the parcel (with the same primary key). Note: because of the uniqueness of parcel identification there will be at most one parcel with the indicated deviceIndex + parceled.
2. By deviceIndex + code: fetches the parcelID related to all parcels of an array with the indicated barcode. If the barcode is unique (it is actually an unambiguous parcel identifier) then only one parcel is identified and returned (info about that parcel can then be fetched using query type 1), otherwise several parcelID values may be returned. Notice that all barcodes associated with a parcel are tried for a match, without any limit of symbology or code group.
3. By code: fetches the deviceIndex and parcelID related to all parcels with the indicated code of any array. One or several parcel may be identified and returned: in particular, even if the input barcode is a unique parcel identifier, multiple results are possible if the same parcel has been processed by several arrays.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).

Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
parcelID	Parcel identifier, as communicated in the EXTENDED_PARCEL message. A fixed length ASCII string.	BIGINT NOT NULL
slotIndex	Index of the code group the barcode belongs to	INT NOT NULL
labelIndex	Position of the barcode in the code group	INT NOT NULL
code	Barcode value	VARCHAR(512) NOT NULL
codeID	AIM code of the symbology of this barcode	VARCHAR(3) NOT NULL
codeLen		INT NOT NULL
numReads	Number of slaves that have read this barcode	INT NOT NULL

Column	Notes	SQL Type
mask	reading-mask	INT NOT NULL
labelX	Y coordinate of barcode	DOUBLE NOT NULL
labelY	Y coordinate of barcode	DOUBLE NOT NULL

Index columns: deviceIndex, parcelId, code

PARCEL_ARCHIVE_SEQ Table

This table contains identification and timestamp (time the EXTENDED_PARCEL message creating a parcel has been received) information about all parcels that have been processed by any array of the plant.
There is one row for each parcel of each array.

This is an alternative the entry point for the search and display of parcels. It can also be used to sequentially (direct and reverse) scanning the parcels processed by an array. Several queries are possible:

1. By deviceIndex + parcelId: fetches the timestamp info related to a specific parcel on an array.
2. By deviceIndex and timestamp: fetches the parcel (its unique key info), with the indicated timestamp, relative to a specific array. Only one result is returned.
3. By deviceIndex and next timestamp: fetches the next parcel (its unique key info), in time, relative to an indicated timestamp and a specific array.
4. By deviceIndex and previous timestamp: fetches the previous parcel (its unique key info), in time, relative to an indicated timestamp and a specific array.
5. By deviceIndex and timestamp range: fetches all parcels (their unique key info) relative to an array and a time interval.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).
Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
parcelID	Parcel identifier, as communicated in the EXTENDED_PARCEL message.	BIGINT NOT NULL
timeStamp	Instant the EXTENDED_PARCEL message creating this parcel has been received	TIMESTAMP

Primary unique key: deviceIndex + parcelID
Index columns: deviceIndex, parcelId, TimeStamp

PARCEL_ARCHIVE_EXT Table

This table contains parcel level information about all parcels that have been processed by any array of the plant.

There is one row for each parcel of each array:

Queries are always by primary key.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).
Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
parcelID	Parcel identifier, as communicated in the EXTENDED_PARCEL message.	BIGINT NOT NULL
parcelLength	It is the triggerOn-TriggerOff distance	DOUBLE NOT NULL
gapFromPrevParcel		DOUBLE NOT NULL
conveyorSpeed		DOUBLE NOT NULL
parcelAnalysis	Value 3 currently not used	INT NOT NULL range = 0..3 0 = GoodRead BaseInterface.rtGood 1 = NoRead BaseInterface.rtNo 2 = MultipleRead BaseInterface.rtMul 3 = PartialRead BaseInterface.rtPar
sideBySide	FALSE means that the parcel is a singulated cuboid	INT NULL = not significant 1 = TRUE 0 = FALSE (singulated cuboid)
parcelSLength	in mm	DOUBLE NULL = not significant
parcelSWidth	in mm	DOUBLE NULL = not significant
parcelSHeigth	in mm	DOUBLE NULL = not significant
parcelSVolume	cm^3	DOUBLE NULL = not significant
parcelXMin	in mm. absolute X coordinate of the leftmost corner of the parcel	DOUBLE NULL = not significant
parcelXMin	in mm. absolute X coordinate of the rightmost corner of the parcel	DOUBLE NULL = not significant

Column	Notes	SQL Type
parcelWeight	In g	DOUBLE NULL = not significant

Primary unique key: deviceIndex + parcelId

PARCEL _ARCHIVE_SLOTEXT Table

This table contains information about all slots that appear all parcels that have been processed by any array of the plant.
There is one row for each slot of each parcel of each array.
Queries are always by primary key

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).

Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
parcelID	Parcel identifier, as communicated in the EXTENDED_PARCEL message.	BIGINT NOT NULL
slotIndex		INT NOT NULL
labelSlot	Identifier of the code group	VARCHAR(10) NOT NULL
numLabel	Number of barcodes of this code group that have been read on this parcel	INT NOT NULL
slotAnalysis	Value 3 not used	INT NOT NULL range = 0..3 0 = GoodRead BaseInterface.rtGood 1 = NoRead BaseInterface.rtNo 2 = MultipleRead BaseInterface.rtMul

Primary non-unique key: deviceIndex + parcelID

PARCEL _ARCHIVE_FILTERCRITEXT Table

This table contains information about all filter rules created.

Write Access: Server Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
CriterionName	Name of the filter criterion	VARCHAR(20) NOT NULL UNIQUE KEY
Criterion	String containing the filter criterion.	CBLOB(100) NOT NULL

primary unique-key CriterionName
This table is only managed by SmartServer (except for its creation)

2.3.9 Operations Configuration

Table OPERATIONS_SETUP provides configuration information about the way the Backoffice and the Server Layers work.
It is a single row table.

Parameter ContinuousUpdate of existing WebSentinel will be dropped since continuous update of windows will no longer be supported.

Write Access: Backoffice Layer: at creation (with default) or restore time.

Server Layer, during system reconfiguration

Read Access: Backoffice Layer.

Server Layer.

Column	Notes	SQL Type
autoSession	Backoffice Layer parameter. Defines whether the session is initiated (and terminated) manually or automatically. N.B.: even if autoSession=1 an operator (with access rights operator) can always stop the current session, so as he can start a session at any time when the system is stopped.	INT NOT NULL 0 = manual session 1 = automatic session (actually no difference between the 2 values: Session is always auto)
sessionHour	Backoffice Layer parameter. Significant even if autoSession=0.	INT NOT NULL Range = 0..23
sessionMin	Backoffice Layer parameter. Significant even if autoSession=0.	INT NOT NULL Range = 0..59
sessionDuration	Backoffice Layer parameter. Significant even if autoSession=0. Expressed in hours, multiple of 1 hour, divides 24 hours, so that automatic sessions are initiated and terminated each day at the same instant.	INT NOT NULL Enumeration = {1, 2, 4, 6, 8, 12, 24}

Column	Notes	SQL Type
lastHourMode	Backoffice & Server Layers parameter.	INT NOT NULL 0 = last hour aligned to session 1 = last hour aligned to clock time
autoConnectOnRestart	Backoffice Layer parameter.	INT NOT NULL 0 = no automatic session start at Backoffice start 1 = automatic session start at Backoffice start
showPreviusSessionData	Server Layer parameter. Controls the presence of Previous Session information fields in the displays.	INT NOT NULL 0 = don't display 1 = display
daysToKeepEdit	Backoffice Layer parameter. Retention time (in days) of report files before they are automatically deleted by the system	INT NOT NULL
exportTXT	Backoffice Layer parameter.	INT NOT NULL 0 = false/No 1 = true/Yes
exportCSV	Backoffice Layer parameter.	INT NOT NULL 0 = false/No 1 = true/Yes
exportXML	Backoffice Layer parameter.	INT NOT NULL 0 = false/No 1 = true/Yes
showLog	Server Layer parameter. Controls whether the Log Window is displayed.	INT NOT NULL 0 = false/No 1 = true/Yes
showParcels	Server Layer parameter. Controls whether Parcel PDUs are displayed in the Log Window	INT NOT NULL 0 = false/No 1 = true/Yes
showPing	Server Layer parameter. Controls whether PingReply PDUs are displayed in the Log Window	INT NOT NULL 0 = false/No 1 = true/Yes
showLastParcel	Server Layer parameter. Controls whether the LastParcel window tab is displayed (whether the window is available to the operator)	INT NOT NULL 0 = false/No 1 = true/Yes

Column	Notes	SQL Type
showEventsLog	Server Layer parameter. Controls whether the EventsLog window tab is displayed (whether the window is available to the operator)	INT NOT NULL 0 = false/No 1 = true/Yes
pingEnabled	Backoffice Layer parameter. To continuously check the connectivity of array controllers. If the functionality is disabled no connection supervision and no automatic reconnection attempts are performed.	INT NOT NULL 0 = false/No 1 = true/Yes
pingRetry	Backoffice Layer parameter. In seconds. The base time interval between one connection attempt to an array and the next one.	INT NOT NULL
pingInterval	Backoffice Layer parameter. In seconds. The time interval between one connectivity check of an array and the next one.	INT NOT NULL
pingTimeout	Backoffice Layer parameter. In seconds. Duration of a connectivity check. If the array's answer is timed out the array is displayed as disconnected.	INT NOT NULL
suppRedundantCA	Backoffice Layer parameter. Defines whether active/standby redundancy is supported, in the form of clearing the ARP table of the WebSentinel machine before any connection attempt with an array.	INT NOT NULL 0 = false/No 1 = true/Yes
updatePeriod	Server Layer parameter. It defines in msec the update period of WebSentinel windows.	INT NOT NULL
logPath	Backoffice Layer parameter. Defines where in the WebSentinel machine file system log, report and backup files are produced. N.B.: log files are automatically generated and retained for a fixed period of **7** days.	VARCHAR(200) NOT NULL

Column	Notes	SQL Type
sessionsToKeep	Number of sessions for which session related history data are kept	INT NOT NULL
daysToKeepAlarms	Number of days for which history alarms are kept	INT NOT NULL
backGndImage	Image to be used as background in the plant Layout window.	VARCHAR(200) NOT NULL
defaultLanguage	Server layer parameter. Defines the default language that will be assigned to a new account when it is defined.	VARCHAR(20) NOT NULL
reportLanguage	Backoffice layer parameter. Defines the language that will be used for reports and SMTP alerts.	VARCHAR(20) NOT NULL
thresholdCrossingColor	Server layer parameter. RGB according to winApi macro	INT NOT NULL
maxSessions	Server layer parameter. Maximum number of simultaneous user sessions that are allowed.	INT NOT NULL
customerLogo	Server layer parameter. Name of file that contains customer logo, if present. NULL if no customer logo (default).	VARCHAR(20)
geniusbridgebtimeout		INT
daysToKeepArrayImagesFile	Number of days for which array image files and related archive DB columns are kept	INT NOT NULL DEFAULT 365

Index columns: -

2.3.10 Current Alarms List

ALARM Table

Table ALARM contains all alarms that are currently active in the Plant.
There is a row for each alarm.

Write Access: Backoffice Layer.

Server Layer: only for lifecycle (in future).

Read Access: Backoffice Layer.

Server Layer.

Column	Notes	SQL Type
deviceIndex	-1 if not relevant (plant alarms)	INT NOT NULL
slaveIndex	-1 if not relevant (array and plant alarms)	INT NOT NULL

Column	Notes	SQL Type
dispName	Name of the resource to which the alarm is related	VARCHAR(20) NOT NULL
probCause		INT NOT NULL
state	Acknowledged or not	INT NOT NULL 0 = unacknowledged 1 = acknowledged
beginDate		TIMESTAMP NOT NULL
category		INT NOT NULL
severity		INT NOT NULL
userName	Identity of user that has acknowledged the alarm (if any)	VARCHAR(20)
comment	Comment introduced by user that has acknowledged the alarm	VARCHAR(100)

Index columns: deviceIndex, slaveIndex, probCause

ALDEVCNT Table

This table contains summary information about the alarms that are currently active in the plant and on each of its components (arrays and slaves).

Alarms related to a plant component are counted also in the hierarchically superior components (an alarm on a slave is counted in the slave, in the containing array and in the plant; an alarm on an array is counted in the array and in the plant).

There will be a set of counters for each plant component and one for the plant as a whole.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).

Server Layer.

Column	Notes	SQL Type
deviceIndex	-1 if not relevant (plant alarms)	INT NOT NULL
slaveIndex	-1 if not relevant (array and plant alarms)	INT NOT NULL
total_s_event	Number of status events	INT
total_warning	Number of alarms with severity = warning	INT
total_minor	Number of alarms with severity = minor	INT
total_major	Number of alarms with severity = major	INT
total_critical	Number of alarms with severity = critical	INT
total_unack	Number of alarms in state unack	INT
total_ack	Number of alarms in state ack	INT

Index columns: deviceIndex, slaveIndex

2.3.11 History Alarms List

Table ALSTO contains all alarms that have occurred in the past in the plant (and that are already cleared).
There is a row for each alarm.

Rows of this table get automatically deleted when they become obsolete (see parameter OPERATIONS.daysToKeepAlarms).

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex	-1 if not relevant (plant alarms)	INT NOT NULL
slaveIndex	-1 if not relevant (array and plant alarms)	INT NOT NULL
dispName	Name of the resource to which the alarm is related	VARCHAR(20) NOT NULL
probCause		INT NOT NULL
state	Acknowledged or not	INT NOT NULL 0 = unacknowledged 1 = acknowledged
beginDate		TIMESTAMP NOT NULL
endDate		TIMESTAMP NOT NULL
category		INT NOT NULL
severity		INT NOT NULL
userName	Identity of user that has acknowledged the alarm (if any)	VARCHAR(20)
comment	Comment introduced by user that has acknowledged the alarm	VARCHAR(100)

Index columns: deviceIndex, slaveIndex

2.3.12 Current Plant Performance Table

Table CSSYSCNT contains the plant level performance information for the current and the preceding sessions and for the last hour.
There are 3 rows, one for the current session, one for the preceding session, and one for the last hour.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).

Server Layer.

Column	Notes	SQL Type
beginSession		TIMESTAMP NOT NULL
parcelCnt		INT NOT NULL

Column	Notes	SQL Type
goodReadCnt		INT NOT NULL
noReadCnt		INT NOT NULL
mulReadCnt		INT NOT NULL
readCnt	Used instead of goodRead when multipleReads counted as goodReads	INT NOT NULL
pcGoodRead		DOUBLE NOT NULL
pcNoRead		DOUBLE NOT NULL
pcMulRead		DOUBLE NOT NULL
pcRead	Used instead of goodRead when multipleReads counted as goodReads	DOUBLE NOT NULL
lowPerf	Low performance condition flag	INT NOT NULL 0 = false 1 = true
curSess		INT NOT NULL 0 = last session 1 = current session 2 = last hour
partialReadCnt		INT NOT NULL
pcPartialRead		DOUBLE NOT NULL

Index columns: curSess

2.3.13 History Plant Performance Table

Table HSSYSCNT contains the plant level performance information for past sessions and hours.
There is one row for each session that is registered.

Sessions are automatically removed when they become stale (see parameter OPERATIONS_SETUP.sessionsToKeep).

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
beginSession		TIMESTAMP NOT NULL
endSession		TIMESTAMP NOT NULL
parcelCnt		INT NOT NULL
goodReadCnt		INT NOT NULL
noReadCnt		INT NOT NULL
mulReadCnt		INT NOT NULL
readCnt	Used instead of goodRead when multipleReads counted as goodReads	INT NOT NULL

Column	Notes	SQL Type
pcGoodRead		DOUBLE NOT NULL
pcNoRead		DOUBLE NOT NULL
pcMulRead		DOUBLE NOT NULL
pcRead	Used instead of goodRead when multipleReads counted as goodReads	DOUBLE NOT NULL
lowPerf	Low performance condition flag	INT NOT NULL 0 = false 1 = true
periodLevel		INT NOT NULL 0 = hour 1 = session
sessionIndex	A unique key associated to the session/hour based on the use of a sequential numbering scheme, so that one can easily move from one session/hour to the next or the preceding one. The session and the hour counter run independently of each other.	INT NOT NULL
partialReadCnt		INT NOT NULL
pcPartialRead		DOUBLE NOT NULL

Index columns: sessionIndex, periodLevel, beginSession
The pair (sessionIndex, periodLevel) represent a primary key of the table.

2.3.14 Current Array Performance Table

CSDEVCNT Table

Table CSDEVCNT contains the array level performance information for the current and the preceding sessions and for the last hour.
There are 3 rows for each array, one for the current session, one for the preceding session, and one for the last hour.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).
Server Layer.

Column	Notes	SQL Type
deviceIndex		INT NOT NULL
parcelCnt		INT NOT NULL
goodReadCnt		INT NOT NULL
noReadCnt		INT NOT NULL
mulReadCnt		INT NOT NULL

Column	Notes	SQL Type
readCnt	Used instead of goodRead when multipleReads counted as goodReads	INT NOT NULL
pcGoodRead		DOUBLE NOT NULL
pcNoRead		DOUBLE NOT NULL
pcMulRead		DOUBLE NOT NULL
pcRead	Used instead of goodRead when multipleReads counted as goodReads	DOUBLE NOT NULL
resetCnt		INT NOT NULL
prcLenAvg	Trigger-on to trigger-off apparent length	DOUBLE NOT NULL
shortPrcCnt		INT NOT NULL
shortGapCnt		INT NOT NULL
pcShortPrc		DOUBLE NOT NULL
pcShortGap		DOUBLE NOT NULL
lostCodCnt	Unassigned codes (out of parcels)	INT NOT NULL
pcLostCod		DOUBLE NOT NULL
gapLenAvg		DOUBLE NOT NULL
speedAvg		DOUBLE NOT NULL
lastSpeed		DOUBLE NOT NULL
lenYAvg	Average Y coordinate of codes. Normally related to the front edge of the parcel, may be absolute depending on array type	DOUBLE NOT NULL
lenXAvg	Average X coordinate of codes. Normally related to the absolute reference system of the array	DOUBLE NOT NULL
readLblCnt	labels counter N.B.: questa e' una informazione di last parcel presente erroneamente in questa tabella	INT NOT NULL
hourDisTime	totalHourDisconnectionTimeInSeconds	DOUBLE NOT NULL
suspectIntervalFlag		INT NOT NULL 0 = false (not suspect) 1 = true (suspect)
lowPerf	Low performance condition flag. If set the color of GoodReadRate window fields is set according to what specified in OPERATIONS_SETUP.thresholdCrossingColor	INT NOT NULL 0 = false 1 = true

Column	Notes	SQL Type
curSess		INT NOT NULL 0 = last session 1 = current session 2 = last hour
partialReadCnt		INT NOT NULL
pcPartialRead		DOUBLE NOT NULL
sideBySideCount		INT NOT NULL
pcSideBySide		DOUBLE NOT NULL
prcLengthAvg		DOUBLE NOT NULL
prcWidthAvg		DOUBLE NOT NULL
prcHeigthAvg		DOUBLE NOT NULL
prcVolAvg		DOUBLE NOT NULL
prcXminPosAvg		DOUBLE NOT NULL
prcXminPosMin		DOUBLE NOT NULL
prcXmaxPosAvg		DOUBLE NOT NULL
prcXmaxPosMax		DOUBLE NOT NULL
prcWeightAvg		DOUBLE NOT NULL

Index columns: deviceIndex, curses

CSSLOTCNT Table

Table CSSLOTCNT contains the array level performance information relative to individual code groups for the current and the preceding sessions and for the last hour.
There are 3xN rows for each array, 3 rows for each code group, one for the current session, one for the preceding session (N may vary for different arrays) and one for the last hour.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).
Server Layer.

Column	Notes	SQL Type
deviceIndex		INT NOT NULL
slotIndex		INT NOT NULL
parcelCnt		INT NOT NULL
goodReadCnt		INT NOT NULL
noReadCnt		INT NOT NULL
mulReadCnt		INT NOT NULL
readCnt	Used instead of goodRead when multipleReads counted as goodReads	INT NOT NULL
pcGoodRead		DOUBLE NOT NULL
pcNoRead		DOUBLE NOT NULL
pcMulRead		DOUBLE NOT NULL

Column	Notes	SQL Type
pcRead	Used instead of goodRead when multipleReads counted as goodReads	DOUBLE NOT NULL
curSess		INT NOT NULL 0 = last session 1 = current session 2 = last hour

Index columns: deviceIndex, slotIndex, curSess

2.3.15 History Array Performance Table

HSDEVCNT Table

Table HSDEVCNT contains the array level performance information for past sessions and hours.
There is one row for each pair (array, session), for each session that is registered.

Sessions are automatically removed when they become stale (see parameter OPERATIONS_SETUP.sessionsToKeep).

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
beginSession		TIMESTAMP NOT NULL
endSession		TIMESTAMP NOT NULL
deviceIndex		INT NOT NULL
parcelCnt		INT NOT NULL
goodReadCnt		INT NOT NULL
noReadCnt		INT NOT NULL
mulReadCnt		INT NOT NULL
readCnt	Used instead of goodRead when multipleReads counted as goodReads	INT NOT NULL
pcGoodRead		DOUBLE NOT NULL
pcNoRead		DOUBLE NOT NULL
pcMulRead		DOUBLE NOT NULL
pcRead	Used instead of goodRead when multipleReads counted as goodReads	DOUBLE NOT NULL
resetCnt		INT NOT NULL
prcLenAvg	Trigger-on to trigger-off apparent length	DOUBLE NOT NULL
shortPrcCnt		INT NOT NULL
shortGapCnt		INT NOT NULL
pcShortPrc		DOUBLE NOT NULL

Column	Notes	SQL Type
pcShortGap		DOUBLE NOT NULL
lostCodCnt		INT NOT NULL
lostPrcCnt		INT NOT NULL
pcLostCod		DOUBLE NOT NULL
gapLenAvg		DOUBLE NOT NULL
speedAvg		DOUBLE NOT NULL
lastSpeed		DOUBLE NOT NULL
lenYAvg	Average Y coordinate of codes. Normally related to the front edge of the parcel, may be absolute depending on array type	DOUBLE NOT NULL
lenXAvg	Average X coordinate of codes. Normally related to the absolute reference system of the array	DOUBLE NOT NULL
readLblCnt	labels counter	INT NOT NULL
hourDisTime	totalHourDisconnectionTimeInSeconds	DOUBLE NOT NULL
suspectIntervalFlag		INT NOT NULL 0 = false (not suspect) 1 = true (suspect)
lowPerf	Low performance condition flag. If set the color of GoodReadRate window field is set according to what specified in OPERATIONS_SETUP.thresholdCrossingColor	INT NOT NULL 0 = false 1 = true
periodLevel		INT NOT NULL 0 = hour 1 = session
sessionIndex	A unique key associated to the session/hour based on the use of a sequential numbering scheme, so that one can easily move from one session/hour to the next or the preceding one. The session and the hour counter run independently of each other.	INT NOT NULL
partialReadCnt		INT NOT NULL
pcPartialRead		DOUBLE NOT NULL
sideBySideCount		INT NOT NULL
pcSideBySide		DOUBLE NOT NULL
prcLengthAvg		DOUBLE NOT NULL
prcWidthAvg		DOUBLE NOT NULL

Column	Notes	SQL Type
prcHeigthAvg		DOUBLE NOT NULL
prcVolAvg		DOUBLE NOT NULL
prcXminPosAvg		DOUBLE NOT NULL
prcXminPosMin		DOUBLE NOT NULL
prcXmaxPosAvg		DOUBLE NOT NULL
prcXmaxPosMax		DOUBLE NOT NULL
prcWeightAvg		DOUBLE NOT NULL

Index columns: deviceIndex, sessionIndex, periodLevel, beginSession
The 3-uple (deviceIndex, sessionIndex, periodLevel) represent a primary key of the table.

HSSLOTCNT Table

Table HSSLOTCNT contains the array level performance information relative to individual code groups for the past sessions and hours.
There is one row for each pair (array, session), for each session that is registered.

Sessions are automatically removed when they become stale (see parameter OPERATIONS_SETUP.sessionsToKeep).

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).

Server Layer.

Column	Notes	SQL Type
deviceIndex		INT NOT NULL
beginSession		TIMESTAMP NOT NULL
endSession		TIMESTAMP NOT NULL
slotIndex		INT NOT NULL
parcelCnt		INT NOT NULL
goodReadCnt		INT NOT NULL
noReadCnt		INT NOT NULL
mulReadCnt		INT NOT NULL
readCnt	Used instead of goodRead when multipleReads counted as goodReads	INT NOT NULL
pcGoodRead		DOUBLE NOT NULL
pcNoRead		DOUBLE NOT NULL
pcMulRead		DOUBLE NOT NULL
pcRead	Used instead of goodRead when multipleReads counted as goodReads	DOUBLE NOT NULL
periodLevel		INT NOT NULL 0 = hour 1 = session

Column	Notes	SQL Type
sessionIndex	A unique key associated to the session/hour based on the use of a sequential numbering scheme, so that one can easily move from one session/hour to the next or the preceding one. The session and the hour counter run independently of each other.	INT NOT NULL

Index columns: deviceIndex, slotIndex, sessionIndex, periodLevel, beginSession
The 4-uple (deviceIndex, slotIndex, sessionIndex, periodLevel) represent a primary key of the table.

2.3.16 Current Slave Performance Table

Table CSSLAVECNT contains the slave level performance information for the current and the preceding sessions and for the last hour.

There are 3 rows for each slave, one for the current session, one for the preceding session, and one for the last hour.

NOTE

In case of a ClusterMultidata array the performance data of a slave node are the same of those of an array.

A different set of counters and percentage indices has been defined for this case.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).

Server Layer.

Column	Notes	SQL Type
deviceIndex		INT NOT NULL
slaveIndex		INT NOT NULL
parcelCnt		INT NOT NULL
goodReadCnt	Applies also to slaves of ClusterMultidata arrays with meaning goodRead	INT NOT NULL
soloReadCnt		INT NOT NULL
pcGoodRead	Applies also to slaves of ClusterMultidata arrays with meaning goodRead	DOUBLE NOT NULL
pcSoloRead		DOUBLE NOT NULL
lowPerf	Low performance condition flag. If set the color of ReadRate window fields is set according to what specified in OPERATIONS_SETUP.thresholdCrossing Color	INT NOT NULL 0 = false 1 = true

Column	Notes	SQL Type
curSess		INT NOT NULL 0 = last session 1 = current session 2 = last hour
noReadCnt	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
mulReadCnt	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
parReadCnt	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
readCnt	Significant in case of a ClusterMultidata array. Defined as = goodRead+parRead+mulRead	INT NOT NULL DEFAULT 0
pcNoRead	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
pcMulRead	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
pcParRead	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
pcRead	Significant in case of a ClusterMultidata array. Defined as = goodRead+parRead+mulRead	INT NOT NULL DEFAULT 0

Index columns: deviceIndex, slaveIndex, curses

2.3.17 History Slave Performance Table

Table HSSLAVECNT contains the slave level performance information for the past sessions and hours.
There is one row for each pair (slave, session), for each session that is registered.

Sessions are automatically removed when they become stale (see parameter OPERATIONS_SETUP.sessionsToKeep).

NOTE

In case of a ClusterMultidata array the performance data of a slave node are the same of those of an array.
A different set of counters and percentage indices has been defined for this case.

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex		INT NOT NULL
slaveIndex		INT NOT NULL

Column	Notes	SQL Type
beginSession		TIMESTAMP NOT NULL
endSession		TIMESTAMP NOT NULL
parcelCnt		INT NOT NULL
goodReadCnt	Applies also to slaves of ClusterMultidata arrays with meaning goodRead	INT NOT NULL
soloReadCnt		INT NOT NULL
pcGoodRead	Applies also to slaves of ClusterMultidata arrays with meaning goodRead	DOUBLE NOT NULL
pcSoloRead		DOUBLE NOT NULL
lowPerf	Low performance condition flag. If set the color of ReadRate window field is set according to what specified in OPERATIONS_SETUP.thresholdCrossingColor	INT NOT NULL 0 = false 1 = true
periodLevel		INT NOT NULL 0 = hour 1 = session
sessionIndex	A unique key associated to the session/hour based on the use of a sequential numbering scheme, so that one can easily move from one session/hour to the next or the preceding one. The session and the hour counter run independently of each other.	INT NOT NULL
noReadCnt	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
mulReadCnt	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
parReadCnt	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
readCnt	Significant in case of a ClusterMultidata array. Defined as = goodRead+parRead+mulRead	INT NOT NULL DEFAULT 0
pcNoRead	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
pcMulRead	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0
pcParRead	Significant in case of a ClusterMultidata array	INT NOT NULL DEFAULT 0

Column	Notes	SQL Type
pcRead	Significant in case of a ClusterMultidata array Defined as = goodRead+parRead+mulRead	INT NOT NULL DEFAULT 0

Index columns: deviceIndex, slaveIndex, sessionIndex, periodLevel, beginSession
The 4-uple (deviceIndex, slaveIndex, sessionIndex, periodLevel) represent a primary key of the table.

2.3.18 SW Inventory Table

This table contains the list of all SW components of the WebSentinel system, including third party components that implement the WebSentinel runtime environment.

For each component the following information is provided:

- Name;
- Version;
- Eventual license/copyright notice.

Table SW_INVENTORY_TABLE will be used to display the SW inventory information in the help-about.
There is a row for each SW component.

Write Access: Backoffice Layer.
Server Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
componentName		VARCHAR(100) NOT NULL
componentVersion		VARCHAR(100) NOT NULL
componentLicense	If applicable, NULL otherwise. Text of license/copyright notice.	VARCHAR(1000)

Index columns: -

2.3.19 Alerts Configuration

These tables contain the configuration info that is necessary to generate spontaneous notification alerts (email notification).
They are written by the Server layer and used by the Backoffice Layer.

Alerts configuration is described by 2 tables:

- ALERTS_SETUP
- ALERTS_TO_SETUP

ALERTS_SETUP Table

This table provides information about the triggering events and the content of email notifications that are automatically generated by WebSentinel.
It is a single row table.

Write Access: Backoffice Layer.
Server Layer.

Read Access: Backoffice Layer.

Column	Notes	SQL Type
smtpEnabled		INT NOT NULL 0 = false/No 1 = true/Yes
sendLowPerfAlarmEnabled		INT NOT NULL 0 = false/No 1 = true/Yes
sendLowPerfAlarm	Significant if sendLowPerfAlarmEnabled=1	INT NOT NULL 1 = send anche alla fine dell'ora 2 = send solo alla fine della sessione
sendDiagAlarmEnable		INT NOT NULL 0 = false/No 1 = true/Yes
tcpDisconnectionTime	in minuti	INT NOT NULL
sendTcpDisconnect		INT NOT NULL range = 0..2 0 = send immediato 1 = send alla fine dell'ora 2 = send alla fine della sessione
sendDeviceDiagnostic		INT NOT NULL range = 0..2 0 = send immediato 1 = send alla fine dell'ora 2 = send alla fine della sessione
sendSlaveDiagnostic		INT NOT NULL range = 0..2 0 = send immediato 1 = send alla fine dell'ora 2 = send alla fine della sessione

Column	Notes	SQL Type
sendNoReadAlarms		INT NOT NULL range = 0..2 0 = send immediato 1 = send alla fine dell'ora 2 = send alla fine della sessione
sendReportOnSessChangeEnabled		INT NOT NULL 0 = false/No 1 = true/Yes
sendPerformanceTXTReportOnSessChange		INT NOT NULL 0 = false/No 1 = true/Yes
sendPerformanceCSVReportOnSessChange		INT NOT NULL 0 = false/No 1 = true/Yes
sendPerformanceXMLReportOnSessChange		INT NOT NULL 0 = false/No 1 = true/Yes
sendAlarmCSVReportOnSessChange		INT NOT NULL 0 = false/No 1 = true/Yes
sendAlarmTXTReportOnSessChange		INT NOT NULL 0 = false/No 1 = true/Yes
smtpServer	A valid IP address of the mail server	VARCHAR(100) NOT NULL
smtpFrom	A valid smtp address: it will result in the sender address of email alerts	VARCHAR(100) NOT NULL

Index columns: -

ALERTS_TO_SETUP Table

This table provides the list of destination of email alerts.
There is a row for each destination of the email alerts.

Write Access: Backoffice Layer.
Server Layer.

Read Access: Backoffice Layer.

Column	Notes	SQL Type

Column	Notes	SQL Type
smtpTo	A valid smtp address. The only check that shoud be performed is syntax (presence of @ and of an IP address or a dns name)	VARCHAR(100) NOT NULL
language	indicates whether the destination is interested in the local default language or in English	INT NOT NULL 0 = English 1 = local default

Index columns: -

2.3.20 Compatibility Setup

Previous versions of WebSentinel supported different versions on the communication protocol.
Additionally, the user may want to configure WebSentinel behaviour, so that it suites his requirements and the requirements of the operating environment.

The COMPATIBILITY_SETUP table provides all this information.
It is a single row table.

Write Access: Backoffice Layer.

Server Layer.

Read Access: Backoffice Layer.

Column	Notes	SQL Type
metric	Relevant for presentation of information by the Server Layer (GUI, save, print) and the Backoffice Layer (report files). The Backoffice layers works always using metric measures.	INT NOT NULL 0 = imperial 1 = metric
comma	Decimal separator character. Relevant for presentation of information by the Server Layer (GUI, save, print) and the Backoffice Layer (report files).	INT NOT NULL 0 = decimal separator “.” 1 = decimal separator “,”
csvSeparator	Relevant for Backoffice Layer. CSV filed separator	VARCHAR(1) NOT NULL
exDiagInfo	Relevant for Backoffice Layer. Speed info available in Diag PDU	INT NOT NULL 0 = not present 1 = present
labelPos	Relevant for Backoffice Layer. Label position info available in Parcel PDU.	INT NOT NULL 0 = not present 1 = present

Index columns: -

2.3.21 Alarms Definition

Table ALMDEFCFG provides configuration information about the alarm causes that are defined in the system. Among the information provided by this table is the default severity of an alarm: each array will redefine this severity, even if the defined value is identical to the default.

If N is the number of alarm causes then table contains N rows.

This table is constructed by the Backoffice layer based on the definition of alarms that is provided by the “Managed Reading System Types” description file: in fact this table exists only logically, since its content can be accessed as part of the “Managed Reading System Types” description file. See paragraph 1.2 for the definition of constants to be used to fill this table.

Write Access: Backoffice Layer.

Read Access: Server Layer.

Backoffice Layer.

Column	Notes	SQL Type
probCause	The acronym of the alarm (probable cause)	VARCHAR(20) NOT NULL
probCauseCode	A numeric identifier of the probable cause	INT NOT NULL
category	The category of the alarm	INT NOT NULL range = 0..6
defaultSeverity	The default severity of the alarm	INT NOT NULL range = 1..4

Index columns: deviceIndex, probCause

2.3.22 Alarms Severity Assignment

Table DEVALMSEVCFG provides configuration information about the severity of alarms, relative to each array.

If N is the number of alarm causes and M the number of arrays in the plant, then table DEVALMSEVCFG contains NxM rows (thus, a row is present for each alarm cause and for each array, even if the severity is equal to the default).

See paragraph 1.2 and paragraph 2.3.21 for the definition of constants to be used to fill this table.
Note: all slaves of an array display assign the same severity to an alarm cause.

Write Access: Backoffice Layer.

Server Layer.

Read Access: Backoffice Layer.

Column	Notes	SQL Type
deviceIndex		INT NOT NULL

Column	Notes	SQL Type
probCause	The numeric identifier of the probable cause (probCauseCode in virtual table ALMDEFCFG)	INT NOT NULL
severity		INT NOT NULL

Index columns: deviceIndex, probCause

2.3.23 Current Events Log

Table CURR_EVENTS_LOG registers all events that are taking place during the session. It is cleared at the beginning of each session and its content is moved to the history events log. Notice that it is possible to show similar info also at array and scanner level.

Write Access: Backoffice Layer.

Read Access: Backoffice Layer (for reporting purposes).

Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
slaveIndex		INT NOT NULL
probCause		INT NOT NULL
severity		INT NOT NULL
category		INT NOT NULL
status	Raise vs. Clear	INT NOT NULL 0 = clear 1 = raise
timeStamp		TIMESTAMP NOT NULL

Index columns: deviceIndex, probCause

2.3.24 History Events Log

Table HIST_EVENTS_LOG registers all events that have taken place during past sessions. History event logs are automatically removed when they become stale (see parameter OPERATIONS_SETUP.sessionsToKeep).

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
slaveIndex		INT NOT NULL
probCause		INT NOT NULL
severity		INT NOT NULL
category		INT NOT NULL

Column	Notes	SQL Type
status	Raise vs. Clear	INT NOT NULL 0 = clear 1 = raise
timeStamp		TIMESTAMP NOT NULL

Index columns: deviceIndex, probCause

2.3.25 Last Parcel Info

Last parcel info is provided through 3 tables:

- table LASTPARCEL_PARCEL
- table LASTPARCEL_SLOT
- table LASTPARCEL_SLOTLABEL

The info provided by these tables is used also to display the “Last Parcel Info” section of the Array.Counters window and the “Last Codes” section of the Slave.Counters window (beside the Last Parcel window).

LASTPARCEL_PARCEL Table

This table contains basic information about the last parcel that has been read by each array of the plant.
There is one row for each array.

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
parcelLength		DOUBLE NOT NULL
gapFromPrevParcel		DOUBLE NOT NULL
conveyorSpeed		DOUBLE NOT NULL
parcelAnalysis	Value 3 currently not used	INT NOT NULL range = 0..3 0 = GoodRead BaseInterface.rtGood 1 = NoRead BaseInterface.rtNo 2 = MultipleRead BaseInterface.rtMul 3 = PartialRead BaseInterface.rtPar

Index columns: deviceIndex

LASTPARCEL_SLOT Table

This table contains information about all slots that appear in the last parcel that has been read by each array of the plant.
There is one row for each slot of each array's last parcel.

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
slotIndex		INT NOT NULL
labelSlot	Identifier of the code group	VARCHAR(10) NOT NULL
numLabel	Number of barcodes of this code group that have been read on this parcel	INT NOT NULL
slotAnalysis	Value 3 not used	INT NOT NULL range = 0..3 0 = GoodRead BaseInterface.rtGood 1 = NoRead BaseInterface.rtNo 2 = MultipleRead BaseInterface.rtMul

Index columns: deviceIndex, slotIndex

LASTPARCEL_SLOTLABEL Table

This table contains information about all barcodes that appear in the last parcel that has been read by each array of the plant.
There is one row for each barcode of each array's last parcel.

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index	INT NOT NULL
slotIndex		INT NOT NULL
labelIndex	Position of the barcode in the code group	INT NOT NULL
code	Barcode value	VARCHAR(512) NOT NULL
codeID	AIM code of the symbology of this barcode	VARCHAR(3) NOT NULL
codeLen		INT NOT NULL

Column	Notes	SQL Type
numReads	Number of slaves that have read this barcode	INT NOT NULL
mask	reading-mask	INT NOT NULL
labelX	Y coordinate of barcode	DOUBLE NOT NULL
labelY	Y coordinate of barcode	DOUBLE NOT NULL

Index columns: deviceIndex, slotIndex

2.3.26 Alarms Propagation and Icon Coloring

These tables provide the Server Layer all information that is necessary to color the plant resource icons that are present in several windows. It also provides the information to colour the digital input icons of the arrays.

Color configuration will take place through a 2 level table system:

- table RESOURCE_COLOR
- table DIGITALINPUT_COLOR
- table COLORS

Table RESOURCE_COLOR will indicate what color different icons should have, but colors are indicated through symbolic values.

Table DIGITALINPUT_COLOR will indicate what color different digital input icons should have, but colors are indicated through symbolic values.

Table COLORS maps symbolic color values to actual colors.

RESOURCE_COLOR Table

There is one row for each plant resource.
Icon columns may be significant or not depending on the resource. When not significant the symbolic color will take value 0.

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex	-1 if not relevant (plant)	INT NOT NULL
slaveIndex	-1 if not relevant (array and plant)	INT NOT NULL
treeIcon	Symbolic color of this icon. Significant also for plant: in tis case it reports only alarms specific of the WebSentinel station.	INT NOT NULL range = 0..4

Column	Notes	SQL Type
treeSlavesIcon	Symbolic color of this icon. Not significant for slaves. Not significant for plant.	INT NOT NULL range = 0..5
layoutIcon	Symbolic color of this icon. Not significant for slaves.	INT NOT NULL range = 0..4
layoutSlavesIcon	Symbolic color of this icon. Not significant for slaves. Not significant for plant.	INT NOT NULL range = 0..5
arraysViewIcon	Symbolic color of this icon. Significant only for arrays.	INT NOT NULL range = 0..4
arraysViewSlavesIcon	Symbolic color of this icon. Significant only for arrays.	INT NOT NULL range = 0..5
scannersViewArrayIcon	Symbolic color of this icon. Not significant for slaves. Not significant for plant. N.B.: Currently unused!	INT NOT NULL range = 0..4
scannersViewSlavesIcon	Symbolic color of this icon. Significant only for slaves.	INT NOT NULL range = 0..5
alarmSummary	Symbolic color of the summary alarm icon to be displayed for arrays and slaves in their summary frame: Colored only if an alarm of the specific resource is currently active.	INT NOT NULL range = 0..5

Index columns: deviceIndex, slaveIndex

DIGITALINPUT_COLOR Table

There is one row for each array.
Each row will have 8+1 columns, corresponding to all possible digital inputs of an array, even though each array may actually handle less than 8 digital inputs.

Write Access: Backoffice Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex		INT NOT NULL
digIn0	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant

Column	Notes	SQL Type
digIn1	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant
digIn2	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant
digIn3	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant
digIn4	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant
digIn5	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant
digIn6	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant
digIn7	Symbolic color of this icon.	INT NOT NULL range = 0..4 0 when not significant

Index columns: deviceIndex

SETUP_COLORS Table

There is one row for each symbolic color. This is a configuration table, which is used for the translation from symbolic colors to actual colors.
There are 5 colors:

1. Red
2. Orange
3. Yellow
4. Green
5. Gray

Write Access: Backoffice Layer.
Server Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
symbolicColor	Array index	INT NOT NULL range = 1.. 5

Column	Notes	SQL Type
actualColor	RGB code (according to winApi macro that combines R, G and B values)	INT NOT NULL

Index columns: deviceIndex, slotIndex

2.3.27 Plant Layout Description

Includes two tables:

- PLANT_LAYOUT
- PLANT_TABS

These tables register the information that is necessary to the Server Layer to draw the Layout window.

PLANT_LAYOUT Table

Write Access: Server Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
deviceIndex	Array index. -1 for plant	INT NOT NULL
upperLeftX	In pixels	INT NOT NULL
upperLeftY	In pixels	INT NOT NULL
horizSize	In pixels	INT NOT NULL
vertSize	In pixels	INT NOT NULL

Index columns: deviceIndex

PLANT_TABS Table

The Layout windows includes several tabs, each of which is described by the following table:

Write Access: Server Layer.

Read Access: Server Layer.

Column	Notes	SQL Type
TABINDEX	Index of the tab	INT NOT NULL
NAME	Name and caption of the tab.	VARCHAR

Index columns: TABINDEX

2.4 CONSISTENCY AND SYNCHRONIZATION

It must be guaranteed that the database doesn't get corrupted even in case of a crash of a client during a write operation.

No explicit synchronization should be performed in the access to the database except it is required for concurrency or resiliency reasons.

3 BACKOFFICE LAYER ARCHITECTURE

The Backoffice Layer is implemented as a single, multi-threaded Java application.

The Backoffice Layer may be in several states, among which:

- **Initializing**: this is the initial state of the Backoffice Layer upon activation: it looks for the existence of the configuration file of a pre-existing WebSentinel installation, and if it finds it, it registers all configuration information in the database.
  While in the Initializing state the Backoffice Layer will not accept the creation of the TCP connection supporting the Server-Backoffice Program Interface.
- **Stopped**: The Backoffice Layer is not connected and is not interacting with any reading station.
- **Started**: The Backoffice Layer is connected or is trying to connect with all reading stations of the plant, and is receiving diagnostic and performance info from all stations it is connected to.

At the end of its Initializing phase the Backoffice Layer enters the Stopped state.
When changing state from Stopped to Started the Backoffice Layer will acquire the configuration of the plant and its own functional configuration from the database.

When changing state from Started to Stopped the Backoffice Layer will produce a backup file of the configuration of the plant and of its own functional configuration from the database.

Just as in the current version, stopping and starting WebSentinel will cause the closure of a session and the opening of a new session.

3.1 DATABASE INITIALIZATION

The database will be initialized at the first startup of the system by the Backoffice Layer that will also populate it either with the information derived from a pre-existing database file or with default values: in any case all required information will be explicitly present in the database.

3.2 SUSPECT INTERVAL FLAG

This flags indicates whether the array has been disconnected for such a long interval that the performance data that have been collected may be not significant.

The computation of the value of this flag is in charge of the Backoffice Layer (both at session and hour level).

Whilst in the past this computation has been based on an absolute disconnection threshold, in WebSentinel it will be based on a percentage threshold (currently not configurable):

- 1/60 of an hour for the last hour flag.
- 1/1440 of a session for the session flag.

The computation of this flag is a responsibility of the Backoffice Layer.

4 EXTERNAL DATABASE ACCESS

Derby database support two kind of connection:

- Local connection (only one local connection can be established)
- Remote connection (based on network server)

When Backoffice starts it builds the database, then makes a local connection and finally starts the network server in order to allow other database connections.

The coordinates for remote database connection are:

Derby User	=	WebSentinelDB
Derby Password	=	WebSentinelDB
Network Server IP	=	localhost
Network Server Port	=	1527

4.1 LOCAL CONNECTION

The local connection is used by Backoffice; it cannot be used by any other process.

4.2 REMOTE CONNECTION

A process can connect to the Derby database using the IBM DB2 JDBC Universal driver or derby client JDBC driver (default is the derby client JDBC driver).

The following java program shows how a process can connect to the WebSentinel database:

```
public class NsSample {
        public static final String DB2_JDBC_UNIVERSAL_DRIVER = new
        String("com.ibm.db2.jcc.DB2Driver");
        public static final String DERBY_CLIENT_DRIVER = "org.apache.derby.jdbc.ClientDriver";
        // network server control specific
        private static int NETWORKSERVER_PORT=1527;
        // Derby database connection URL for embedded environment
        public static final String CS_EMBED_DBURL="jdbc:derby:NSSampledb;";

        // To connect to Derby Network Server
        // This URL describes the target database
        // Notice that the properties may be established via the URL syntax
        private static final String CS_NS_DBURL=

        "jdbc:derby:net://localhost:"+NETWORKSERVER_PORT+"/WebSentinelDB;create=true;retrie
        veMessagesFromServerOnGetMessage=true;deferPrepares=true;";
```

```
 // URL for the Derby client JDBC driver.
private static final String DERBY_CLIENT_URL=
"jdbc:derby://localhost:"+NETWORKSERVER_PORT+"/WebSentinelDB;create=true;";
// Default to using the Derby Client JDBC Driver for database connections
String url = DERBY_CLIENT_URL;
String jdbcDriver = DERBY_CLIENT_DRIVER;

public static void main(String[] args) throws Exception  {
        new nserverdemo.NsSample().startSample(args);
} // main

public void startSample(String[] args) throws Exception {
        NetworkServerUtil nwServer;
        Connection conn = null;
        PrintWriter pw = null;
        // Load the JDBC Driver
        try {
           Class.forName(jdbcDriver).newInstance();
        } catch (Exception e) {
            pw.println("[NsSample] Unable to load the JDBC driver. Following exception was
thrown");
            e.printStackTrace();
            System.exit(1);  //critical error, so exit
        }
        // See Derby documentation for description of properties that may be set
        //  in the context of the network server.
        Properties properties = new java.util.Properties();
        // The user and password properties are a must, required by JCC
        properties.setProperty("user","WebSentinelDB");
        properties.setProperty("password","WebSentinelDB");

        // Get database connection via DriverManager api
        try {
           conn =  (Connection) DriverManager.getConnection(url, properties);
        } catch(Exception e) {
           pw.println("[NsSample] Connection request unsuccessful, exception thrown was:
");
           pw.println("[NsSample] Please check if all the jar files are in the classpath and the
dbUrl is set correctly.");
           e.printStackTrace();
           System.exit(1);  //critical error, so exit
```

```
                }
                //……TO BE DONE
                 conn.close();
                //……TO BE DONE
                 } catch (Exception e) {
                        e.printStackTrace();
                }
        } // startSample

        /**
          * Determine which jdbc driver to use by parsing the command line args.
          *  Accepted values:
          *  jccjdbclient   - The DB2 type 4 universal driver
          *  derbyclient    - The Derby network driver (default).
          *  Note: because this is just a sample, we only care about whether
          *  the above values are specified.  If they are not, then we default
          *  to the Derby network driver.
        */
        private void parseArguments(String[] args)
                    {
                       int length = args.length;
                       for (int index = 0; index < length; index++)
                       {
                           if (args[index].equalsIgnoreCase("jccjdbcclient"))
                           {
                              jdbcDriver = DB2_JDBC_UNIVERSAL_DRIVER;
                              url = CS_NS_DBURL;
                              break;
                             } else if (args[index].equalsIgnoreCase("derbyClient"))
                           {
                              jdbcDriver = DERBY_CLIENT_DRIVER;
                              url = DERBY_CLIENT_URL;
                              break;
                           }
                       }
                    }// parseArguments
}// NsSample
```

4.3 SQL SAMPLES

The following query retrieves all the arrays of the plant.

```
SELECT * FROM S.DEVICECFG ORDER BY deviceIndex;
```

The following query retrieves an array using its name (“array1”)

```
SELECT * FROM S.DEVICECFG WHERE name=“array1”;
```

The following two queries retrieve all the readers of a given array

- using the array name:

```
SELECT slave.* FROM S.SLAVECFG slave
        WHERE slave.deviceIndex  IN
                SELECT array.deviceIndex
                        FROM S.DEVICECFG array
                                WHERE array.name=”array1”;
```

- using IP address and Port of the array

```
SELECT slave.* FROM S.SLAVECFG slave
        WHERE slave.deviceIndex  =
                SELECT array.deviceIndex
                        FROM S.DEVICECFG array
                                WHERE array.addr=”171.16.11.01” AND
                                        array.port = 51232;
```

The following query retrieves the active alarms of the plant

```
SELECT al.* FROM S.ALARM al
        WHERE al.deviceIndex=-1 AND al.slaveIndex=-1;
```

The following query retrieves the active alarms of all arrays of the plant:

```
SELECT al.* FROM S.ALARM al
        WHERE al.deviceIndex<>-1 AND al.slaveIndex=-1;
```

The following query retrieves the active alarms of all readers of the plant:

```
SELECT al.* FROM S.ALARM al
        WHERE al.deviceIndex<>-1 AND al.slaveIndex<>-1;
```

The following query retrieves the active alarms of a given array of the plant:

```
SELECT al.* FROM S.ALARM al
        WHERE al.deviceIndex =
                (SELECT array.deviceIndex
                        FROM S.DEVICECFG array
                                WHERE array.addr=”171.16.11.01” AND
                                        array.port = 51232)
                AND al.slaveIndex=-1;
```

The following query retrieves the active alarms of all readers of a given array of the plant:

```
SELECT al.* FROM S.ALARM al
        WHERE al.deviceIndex =
                (SELECT array.deviceIndex
                        FROM S.DEVICECFG array
                                WHERE array.addr=”171.16.11.01” AND
                                array.port = 51232)
                AND al.slaveIndex<>-1;
```

5 MEANINGLESS FIELDS IN ON-LINE MODE

Counters Window

- Conveyor Speed
- Last Parcel Info - X:Y Position
- Last Parcel Info - Length
- Last Parcel Info – Gap
- Session Statistics – Short Parcels
- Session Statistics – Short Gaps
- Session Statistics – Lost Codes
- Session Statistics – Average (Avg.) Parcel Length
- Session Statistics – Average (Avg.) Gap Length
- Session Statistics – Average (Avg.) Conveyor Speed
- Session Statistics – Average (Avg.) X:Y Position
- Last Hour Statistics – Average (Avg.) Short Parcels
- Last Hour Statistics – Short Gaps
- Last Hour Statistics – Lost Codes
- Last Hour Statistics – Average (Avg.) Parcel Length
- Last Hour Statistics – Average (Avg.) Gap Length
- Last Hour Statistics – Average (Avg.) Conveyor Speed
- Last Hour Statistics – Average (Avg.) X:Y Position