
FORTRA
IBM i Security Auditing
Deep Dive

Presenter

FORTRA

First encounter with AS/400 in 1995

Bonn, Germany

Senior

Kurt Thomas, CISSP

Senior System Engineer
Fortra

2

Willkommen, bienvenue, welcome!

FORTRA

I promise that you will learn things about the IBM i security audit journal that you did not know:

- How it works
- Why you need it
- How you can use it

Deep Dive—let *me* do the work!

We will travel back in time all the way to the 1970s—and we're gonna go 4-dimensional!

3

Willkommen, bienvenue, welcome!

FORTRA

This Deep Dive is aimed at IBM i admins, compliance officers, and people of varying levels of knowledge of IBM i

Enablement: My goal is to give you enough knowledge for you to start experimenting on your own

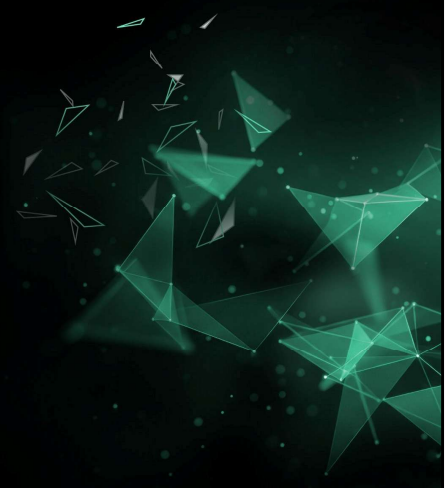
All recommendations are based on experience (mine and others')

But first: Some …

4

Rules!

Do ask me questions
We will take breaks



5

I will answer your questions in this session if time allows, else offline

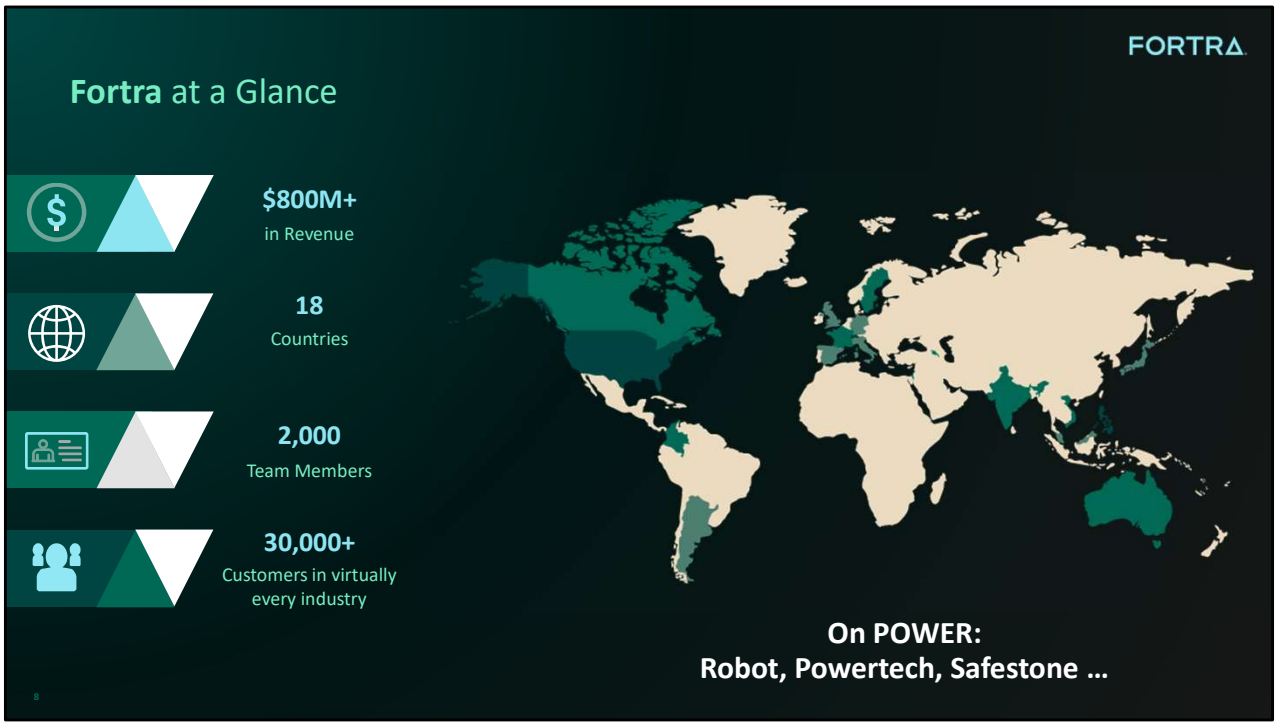
For Our French Participants

Hé
Viens près de moi que je te le dise
Ce secret qui me tord le cœur

6

I will answer your questions in this session if time allows, else offline


FORTRA
About Fortra

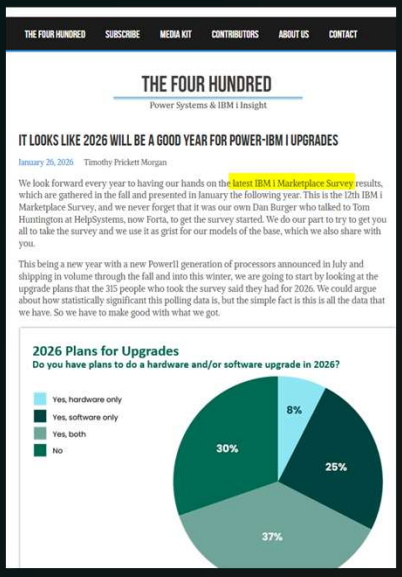


Our products are amongst the **most widely-used solutions on IBM i and IBM Power**
You know our brands: **Robot**, **Powertech**, Safestone, …
Security, automation, systems management, document management, business intelligence
We also develop a number of IBM i solutions that IBM markets
Have you heard of BRMS, PowerHA, or Rational IDE for IBM i?

FORTRA

Fortra: Taking the Pulse of IBM i

- Fortra publishes the annual **State of IBM i Security Study**
  - Specific to IBM i security
  - Based on actual system configuration data
- Fortra runs the Annual **IBM i Marketplace Survey**
- Both the **Survey** and the **Study** are regularly cited by IBM and by media like *IT Jungle*



THE FOUR HUNDRED SUBSCRIBE MEDIA KIT CONTRIBUTORS ABOUT US CONTACT

THE FOUR HUNDRED

Power Systems & IBM i Insight

IT LOOKS LIKE 2026 WILL BE A GOOD YEAR FOR POWER-IBM I UPGRADES

January 26, 2026 Timothy Prickett Morgan

We look forward every year to having our hands on the latest IBM i Marketplace Survey results, which are gathered in the fall and presented in January the following year. This is the 12th IBM i Marketplace Survey, and we never forget that it was our own Dan Burger who talked to Tom Huntington at HelpSystems, now Forta, to get the survey started. We do our part to try to get you all to take the survey and we use it as grist for our models of the base, which we also share with you.

This being a new year with a new Power11 generation of processors announced in July and shipping in volume through the fall and into this winter, we are going to start by looking at the upgrade plans that the 315 people who took the survey said they had for 2026. We could argue about how statistically significant this polling data is, but the simple fact is this is all the data that we have. So we have to make good with what we got.

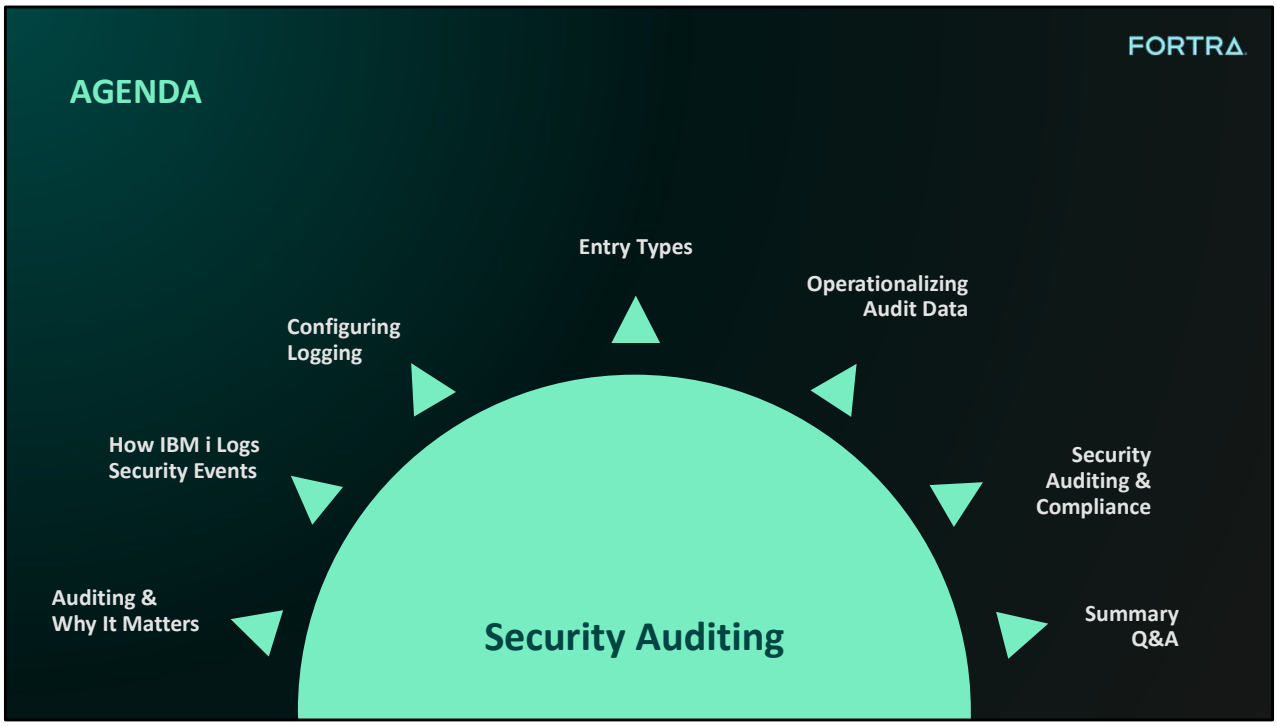
2026 Plans for Upgrades

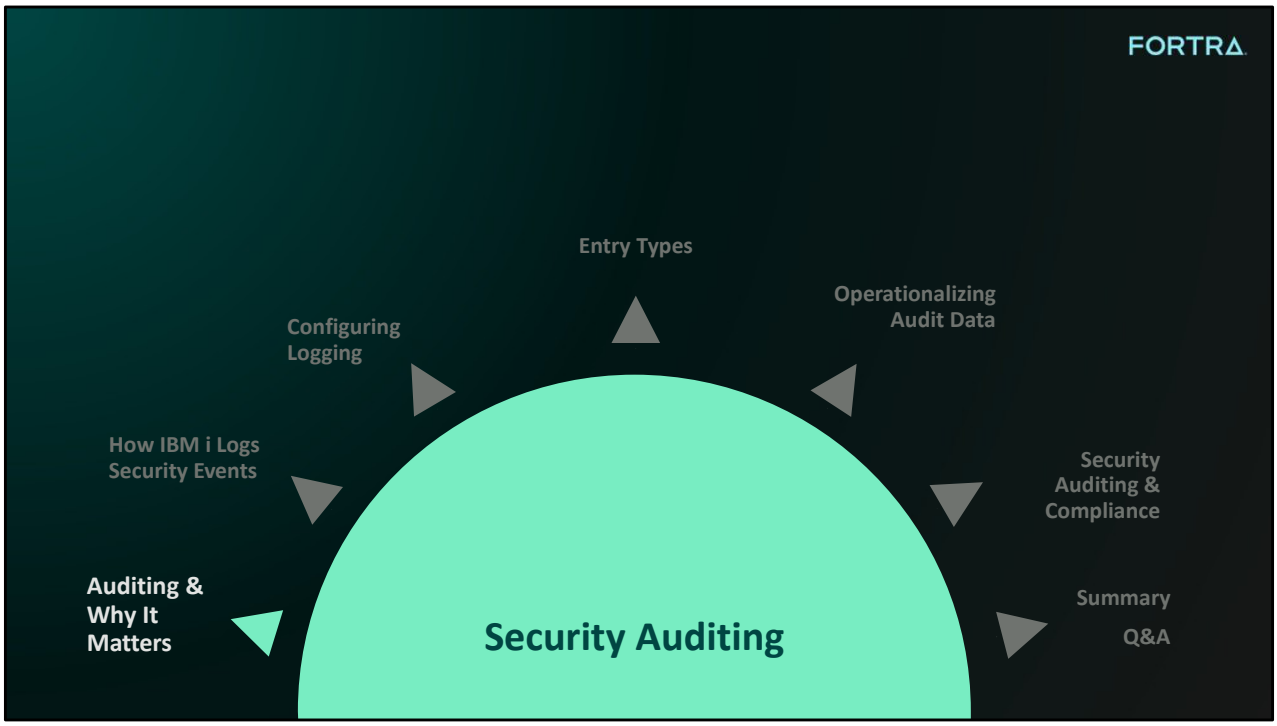
Do you have plans to do a hardware and/or software upgrade in 2026?

- Our Infrastructure Protection Suite, consisting of our Vulnerability Management and Offensive Security solutions allow you to:
    - Identify and prioritize vulnerabilities in your systems
    - Understand which of these vulnerabilities are exploitable
    - Conduct Red team exercises to understand how an attacker could exploit these vulnerabilities and how best to respond to them
- Our Data Security Suite, consisting of our Data Loss Prevention, Digital Risk Protection, Secure File Transfer and Email Security solutions allow you to:
    - Identify your sensitive data so that you can wrap the appropriate security controls around it, without becoming a barrier to legitimate business processes
    - Ensure that when it is being shared appropriately, it is being shared securely
    - Protect your sensitive data from external threats, such as ransomware, spyware and phishing
- Our Threat Research and Intelligence team enhance our host of threat feeds through researching and augmenting our data collection. Additionally, they leverage our Threat Fusion Center data to find new threats that may have been

missed by automated classifiers or operations analysts; as well as investigate confirmed threats to analyze trends or identify attackers.

- Our Automation solutions allow you to:
    - Boost productivity, improve accuracy, and help your organization grow by handing repetitive, manual workloads
    - Provide non-technical users with the means to automate tasks, while also allowing highly technical users to automate complex business processes, such as batch processing
    - Free up valuable IT resources by automating time consuming tasks, such as manual monitoring, optimization and so on
- Our Centralized Analytics solutions:
    - Simplify product administration
    - Centralize key metrics
    - Provide mobile control
    - Our new Fortra One platform provides dashboards for security and operations data, helps customers deploy our software, many products have user interface access from here and you can even do Analytics from this panel.


FORTRA
AGENDA
Entry Types
Operationalizing Audit Data
Configuring Logging
How IBM i Logs Security Events
Security Auditing & Compliance
Auditing & Why It Matters
Summary Q&A
Security Auditing


FORTRA
Entry Types
Operationalizing Audit Data
Configuring Logging
How IBM i Logs Security Events
Security Auditing & Compliance
Auditing & Why It Matters
Summary
Q&A
Security Auditing

Security Auditing and Why It Matters

- What I mean when I say “security auditing”
- What are “security events”?
- Compliance pressure
- Why auditing matters to you

What I Mean When I Say “Security Auditing”

Security Auditing and Why It Matters

- In this Deep Dive, when I say “security auditing”, I mean the total of:
  - Configuring your IBM i to log security events
  - IBM i logging those events
  - Retrieving the logged events and …
  - … Using that information to:
    - Document what events occurred on your system
    - Investigate security issues or have them be investigated
    - Satisfy demands of an auditor

This is how I use the term “security auditing” here. IBM in its own documentation uses the term differently, with at least two meanings:

- checking what happens on your system
- logging security-relevant events, e.g. “an AF type entry is written to the audit journal if the auditing function is active”

What Are “Security Events”?

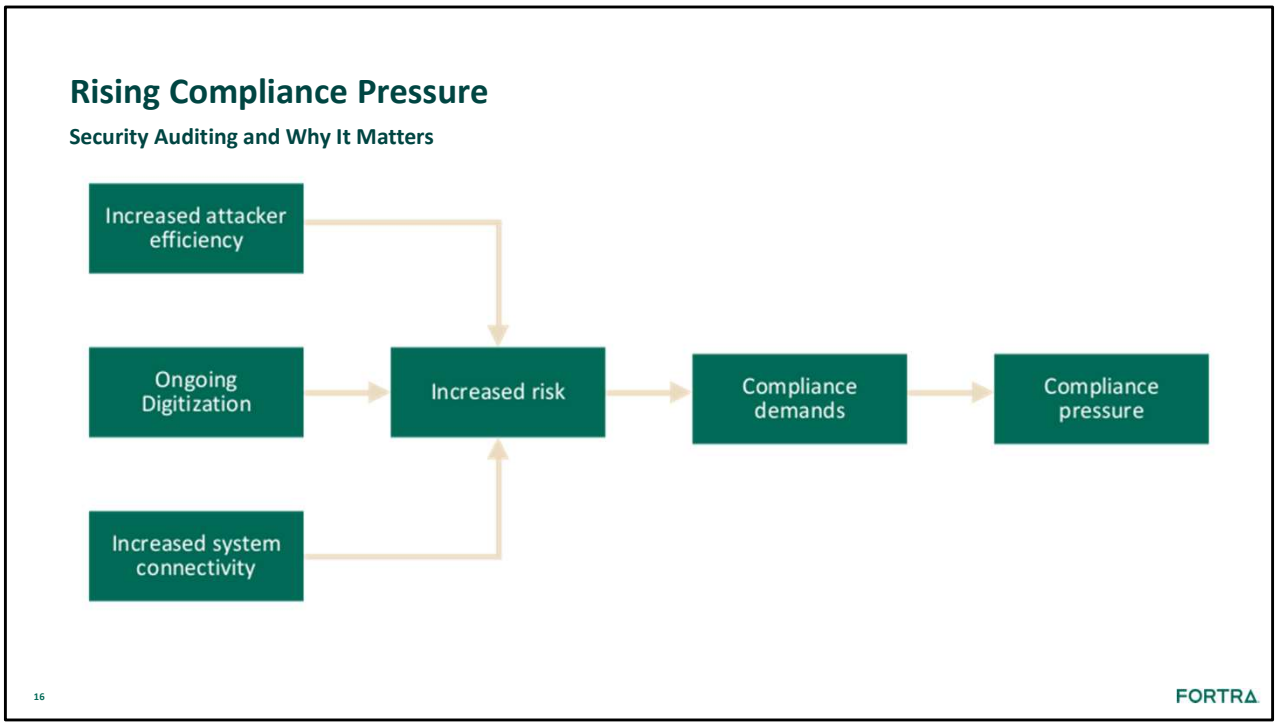
Security Auditing and Why It Matters

- Security events are events that:
  - Change the security posture of your system in any way
  - Have the potential for data to be leaked, or
  - Are attempts to violate security settings
- Additionally, they include events that could be useful in forensic reconstructions of events
- In practical terms, security events are the kinds of events that **IBM has decided are worth logging in the audit journal**

FORTRA

Some examples are:

Adding/changing/deleting user profiles
Changing the *PUBLIC authority of a streamfile
Changing system values, including the current time
Job start and termination
Authentication failures
Communication between jobs that does not use objects



This is a simplified picture of trends influencing what I call compliance pressure—the pressure to do things in order to be compliant with a regulatory or industry framework.
Here we are looking at security compliance.

This is simplified in that some nodes and edges are omitted for clarity. Other factors contribute to the compliance pressure, and factors interact. Increased attacker efficiency has ongoing digitization as one root, too, but here I focus on digitization for the potentially attacked organizations. “Software is eating the world”.

We will revisit this topic later in the *Compliance Frameworks* section.

Why It Matters to You

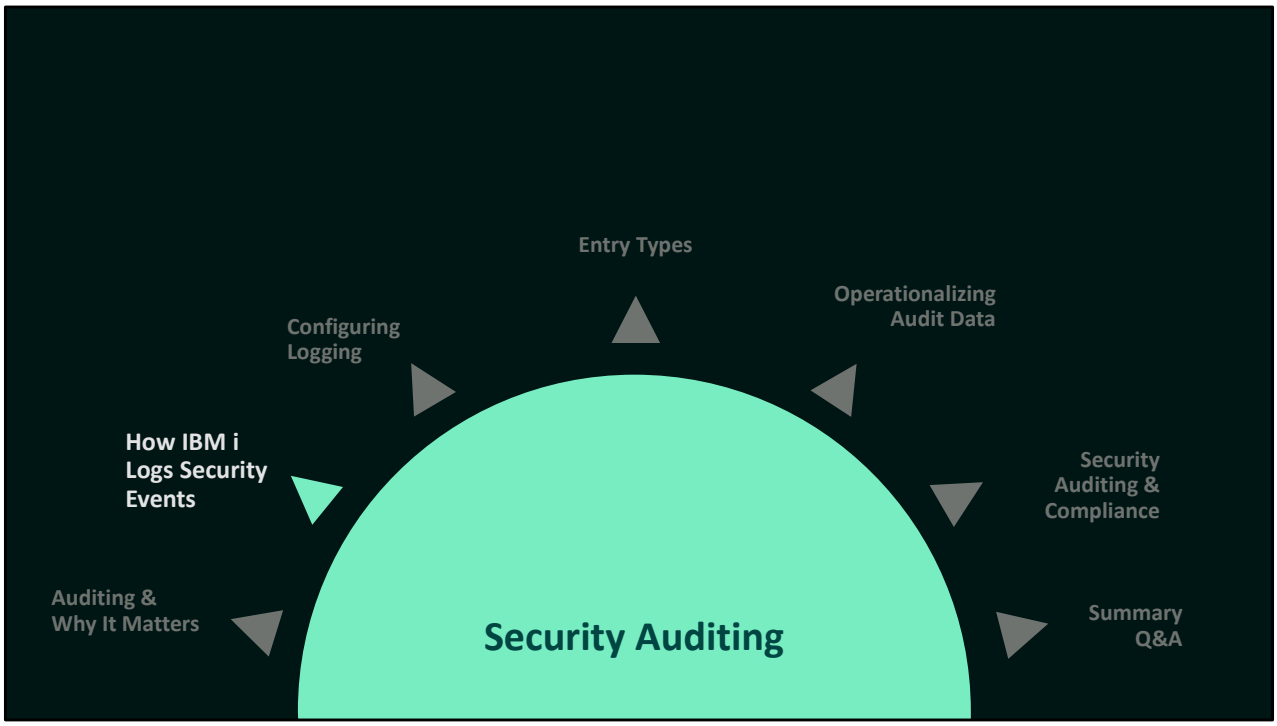
Security Auditing and Why It Matters

- Attacks on security are becoming more pervasive
- Even the IBM i can be affected, and IBM i systems are business-critical
- Auditors want these events
- Events are useful in for investigating security issues (forensic)
- The audit journal == the single most important source of security event data on IBM i

Summary: Security Auditing and Why It Matters to You

Security Auditing and Why It Matters

- Security auditing here means the writing, reading and using of IBM i audit journal entries that represent security-relevant events
- The importance of security auditing increases in proportion to cyber risks and compliance demands
- Both security risk and compliance demands are rising
- The security audit journal is the central source of security event information on IBM i


Entry Types
Operationalizing Audit Data
Configuring Logging
How IBM i Logs Security Events
Security Auditing & Compliance
Auditing & Why It Matters
Summary Q&A
Security Auditing

How IBM i Logs Security Events

- The security audit journal
- Where do events come from?
- Case study
- Why a journal
- The header
- The data section
- Changes between versions

The Security Audit Journal

How IBM i Logs Security Events

- On IBM i, the security audit journal is the **central log of security events**
- If you ever have to explain the audit journal to Microsoft Windows folks, say:

"It's like the Security Event Log on Windows"

Where Do Events Come From?

How IBM i Logs Security Events

Clue:

- The logging of security events does NOT require application cooperation

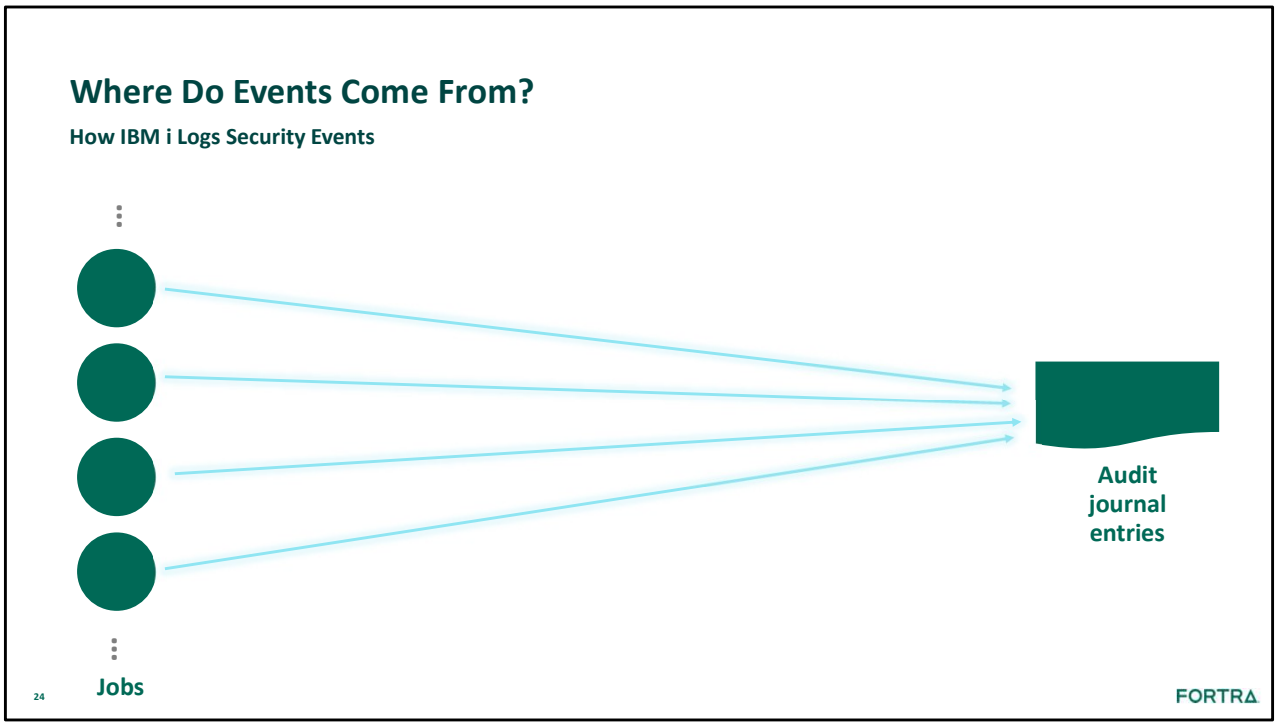
FORTRA

When audit entries are created, that is not because the developer of the application said: “We will log these events now”. The entries are created with or without the developer’s consent.

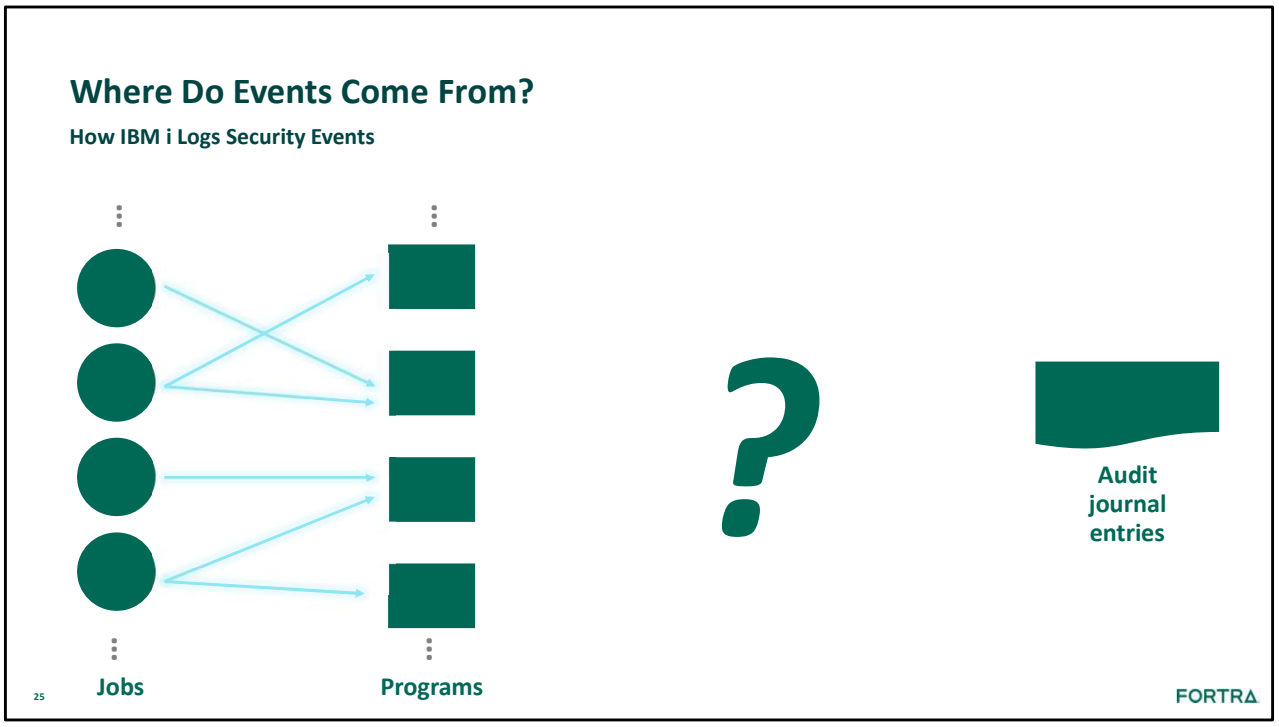
Where Do Events Come From?

How IBM i Logs Security Events

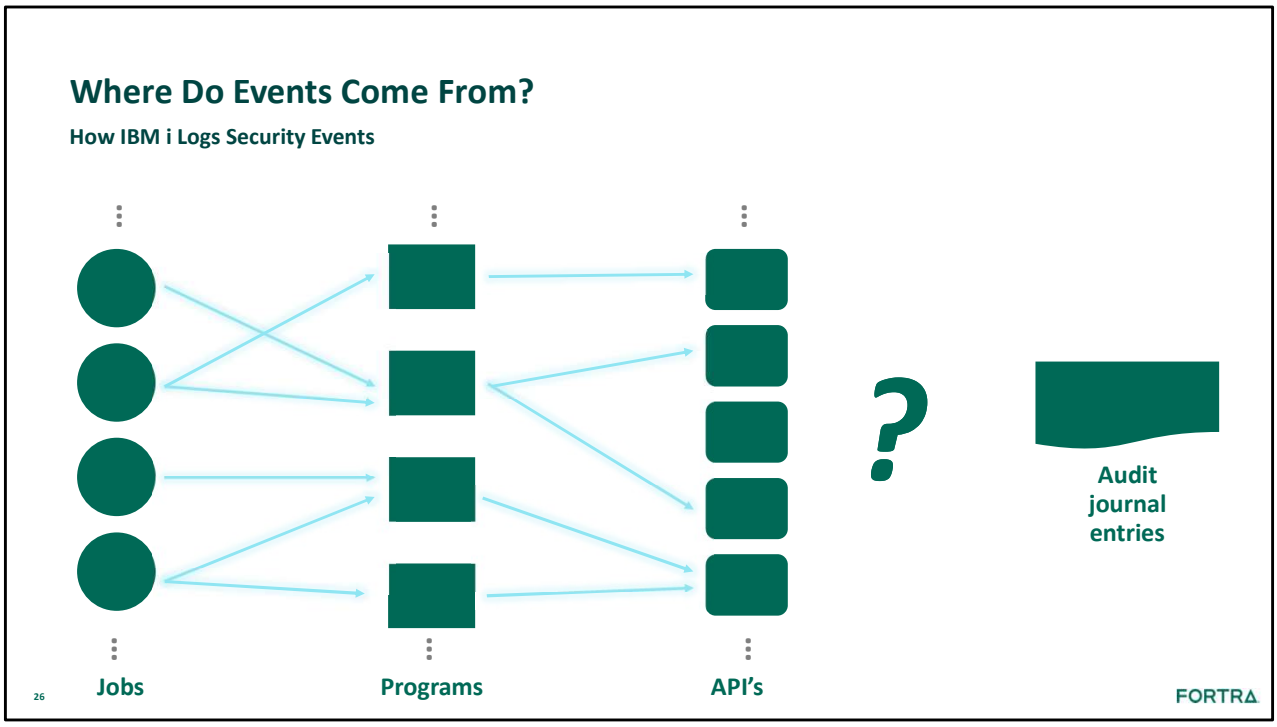
FORTRA



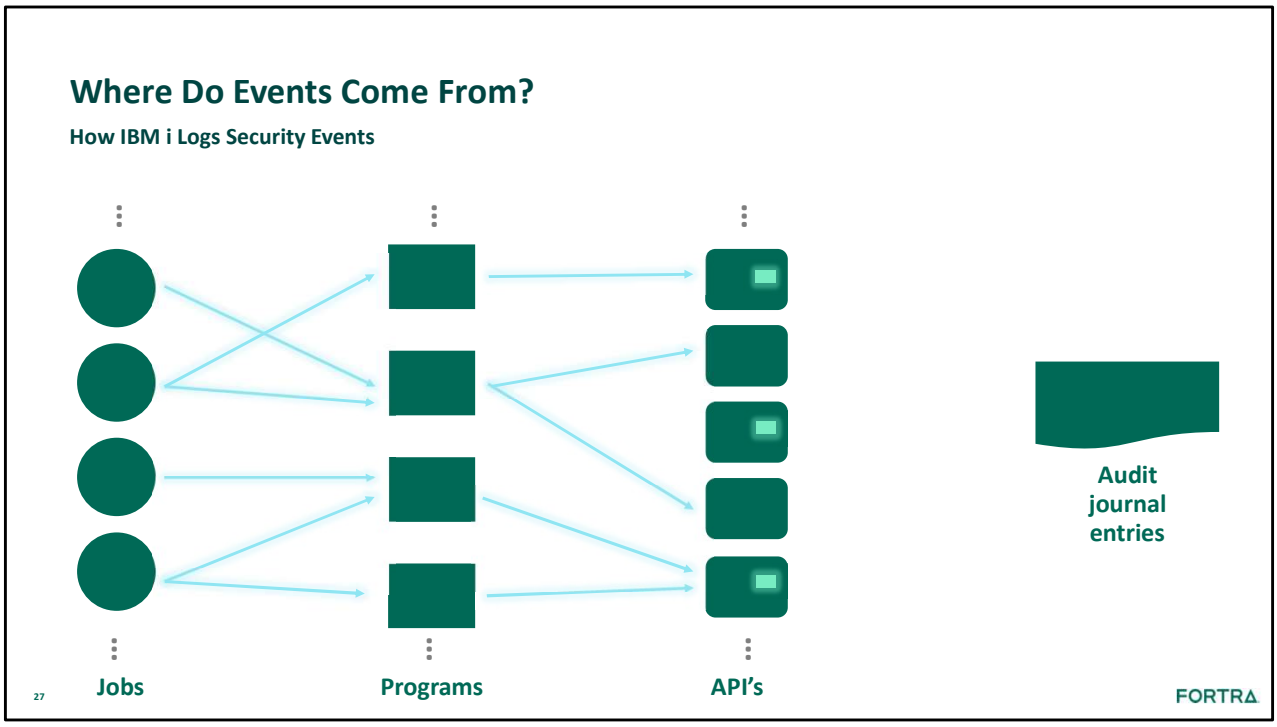
For the most part, things that happen in IBM i happen in **jobs**.
(Only occasionally is an audit journal entry about something that happened outside of a job. It then is an event which the Licensed Internal Code asks the job to log.)
Well, what is a job?



A job is a logical container in which programs are executed.
OK, so far so good. So, do the programs create the audit journal entries? No, or at least not those programs that make up the applications executables.
The developer of the application controls which programs are run.
Do we want to leave it to the application developer whether to log security events?
Hell, no!
The OS has to enforce the logging.
So how can that be done?
Well all programs that are run in a job eventually call …

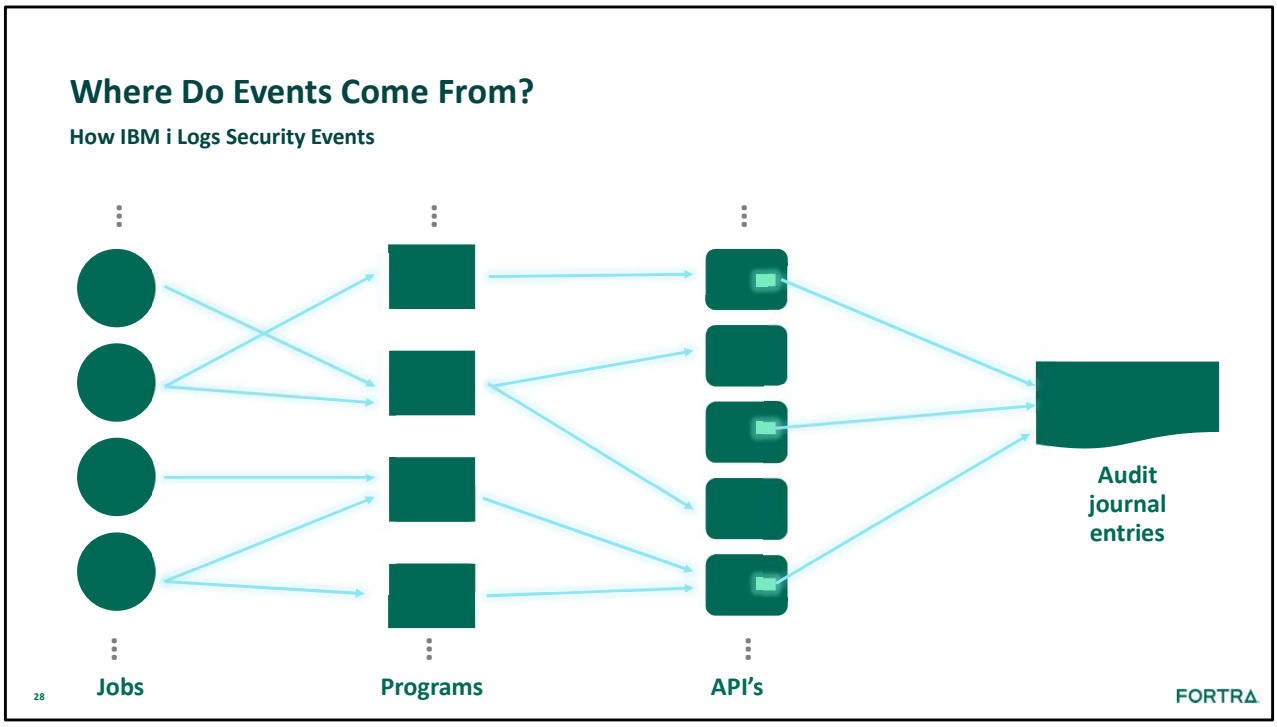


API's, and the **API's are under control from IBM**. So what IBM did was to say:



"This API does something worth logging, so we will put something into the API and if that API is used, we log the API's activity"

IBM *instrumented* the API's to ensure that the logging will happen



So that then closes the loop how we go from "something is happening on the system" to "an audit journal entry is written".
This logging is baked in at a low level. And because applications cannot directly call the kernel --- only IBM-created software can--it would very, very hard to circumvent this.

Where Do Events Come From?

How IBM i Logs Security Events

29

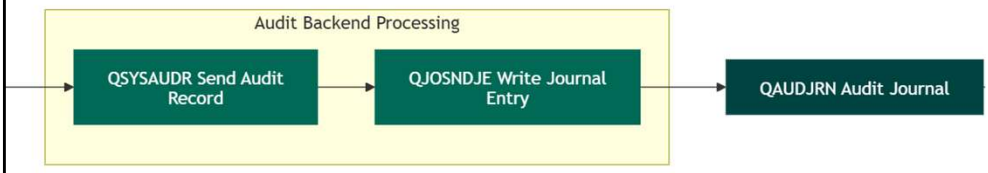
FORTRA

For a single job, this is the sequence:
1: A job executes programs, which eventually call IBM APIs
2: A called API was instrumented —by IBM—to create an audit event. Let's call this an audit "hook". The hook kicks off the log writing process. "Hey, I am doing …. Service Tools stuff, I will write a journal entry that says I did that"
4: The job determines if the event needs to be logged for this particular invocation of the API. It already has the hard-coded info that an event of a specific type—say, ST—can be logged. It now looks at other factors to make the final decision. We will talk about these factors, too.
5: If yes, the action should be loged, hen the data for the audit journal entry is prepared—he payload.
6: The audit journal entry is written

Where Do Events Come From?

How IBM i Logs Security Events

The final action of writing the audit journal entry uses the same programs across entry types



30

FORTRA

When we talk about creating the journal entry, these two programs are always used, for any audit journal entry type.
First, QSYSAUDR is the final audit journal specific program in the sequence. It does the final preparation of data for the writing of the entry. This is all of the variable data and some of the fixed-header data. part of the entry.
Then it passes its little package to QJOSENDJE. QJOSENDJE is a general program for writing a journal entry. It is used by anything that creates a journal entry. It is not specific to audit journal. That then does the hard work. Well, actually, it hands off the hard work to a whole other hierarchy of programs in the Licensed Internal Code, but that is not our problem here-
The result is an audit journal entry.

Where Do Events Come From?

How IBM i Logs Security Events

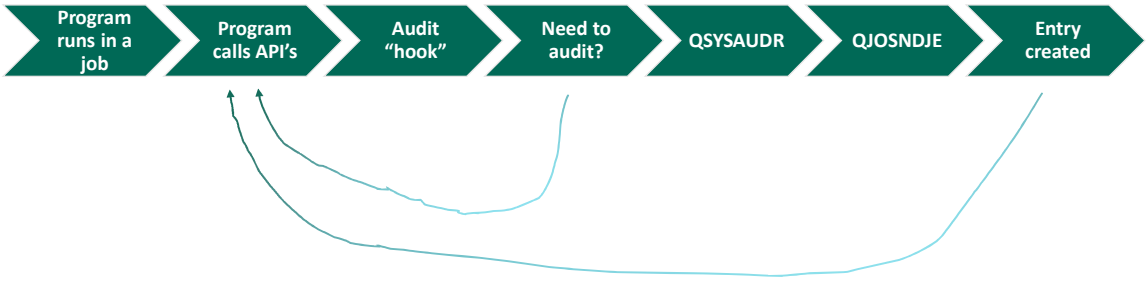
31

FORTRA

So, slight refinement of our previous picture here …

Where Do Events Come From?

How IBM i Logs Security Events



32

FORTRA

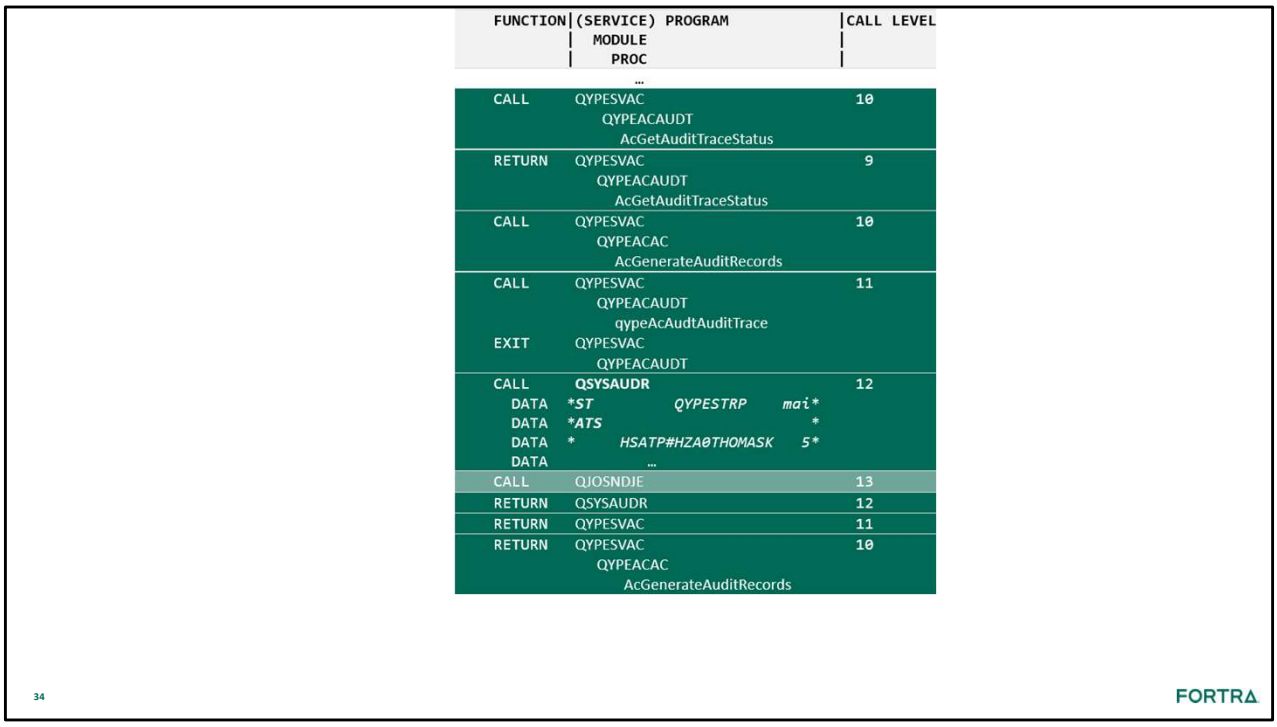
… to include QSYSADUR and QJOSNDJE.

After the audit part is done, either because there was no logging to do, or because the journal entry has been logged, the execution flow returns to the API and whatever it does.

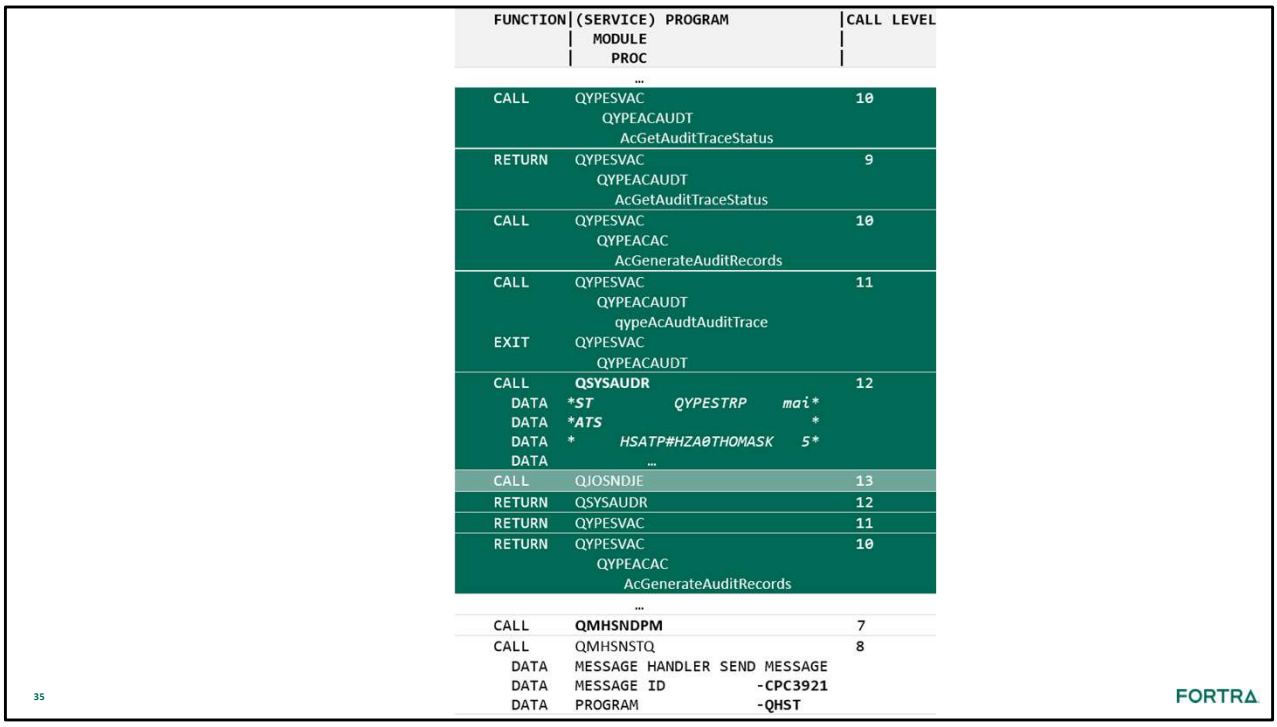
Case Study: A Job Trace Is Started

How IBM i Logs Security Events

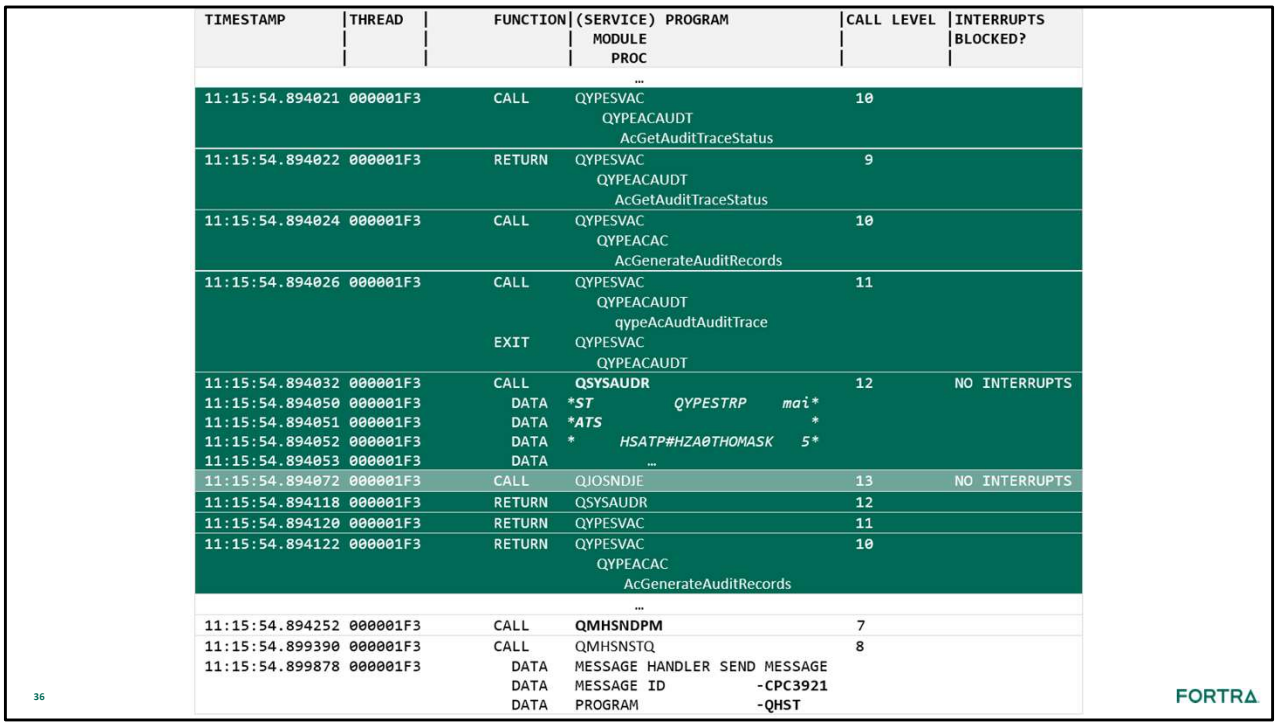
- Scenario: The user starts a job trace
- How is the audit journal entry created?
- The following shows an excerpt from the resulting job trace
- The auditing-aware stuff is in green
- Starting a trace logs an ST = “Service Tools Usage” entry



- First it checks if Service Tools Usage is audited
- The answer is “Yes”
- Then we prepare the data to be send to the journal entry, in the format for entry type ST = Use of Service Tools
- QSYSAUDR is the central API to log an audit journal entry. It receives most of the data for the audit journal entry from the programs that calls QSYSAUDR
- Data: ST => “Use of a Service Tool”, A => the only subtype for ST entries; “TS” => what service tool was used: STRTRC
- QSYSAUDR then uses the general QJSDNJE API to create a journal entry. That API is used for all journal entry writing, not just for audit journals.
- Note thread ID stays identical: Single threaded – “Focus on one thing”
- Interrupts are blocked during critical action—write of journal entry must be all-or-nothing
- Only *after* audit entry is logged can other action be taken, e.g. send a message to say trace has started
- Very fast: Total time used on audit-related things is about one tenth of a millisecond on a POWER8 server
- Sending an entry to the audit journal and a message to the history log is typical of a mixed, “organically grown” auditing approach



- First it checks if Service Tools Usage is audited
- The answer is “Yes”
- Then we prepare the data to be send to the journal entry, in the format for entry type ST = Use of Service Tools
- QSYSAUDR is the central API to log an audit journal entry. It receives most of the data for the audit journal entry from the programs that calls QSYSAUDR
- Data: ST => “Use of a Service Tool”, A => the only subtype for ST entries; “TS” => what service tool was used: STRTRC
- QSYSAUDR then uses the general QJSDNJE API to create a journal entry. That API is used for all journal entry writing, not just for audit journals.
- Note thread ID stays identical: Single threaded – “Focus on one thing”
- Interrupts are blocked during critical action—write of journal entry must be all-or-nothing
- Only *after* audit entry is logged can other action be taken, e.g. send a message to say trace has started
- Very fast: Total time used on audit-related things is about one tenth of a millisecond on a POWER8 server
- Sending an entry to the audit journal and a message to the history log is typical of a mixed, “organically grown” auditing approach



TIMESTAMP	THREAD	FUNCTION	(SERVICE) PROGRAM MODULE PROC	CALL LEVEL	INTERRUPTS BLOCKED?
			…		
11:15:54.894021	000001F3	CALL	QYPESVAC QYPEACAUDT AcGetAuditTraceStatus	10	
11:15:54.894022	000001F3	RETURN	QYPESVAC QYPEACAUDT AcGetAuditTraceStatus	9	
11:15:54.894024	000001F3	CALL	QYPESVAC QYPEACAC AcGenerateAuditRecords	10	
11:15:54.894026	000001F3	CALL	QYPESVAC QYPEACAUDT qypeAcAudtAuditTrace	11	
		EXIT	QYPESVAC QYPEACAUDT		
11:15:54.894032	000001F3	CALL	**QSYSAUDR**	12	NO INTERRUPTS
11:15:54.894050	000001F3	DATA	*ST QYPESTRP mai*		
11:15:54.894051	000001F3	DATA	*ATS *		
11:15:54.894052	000001F3	DATA	* HSATP#HZA0THOMASK 5*		
11:15:54.894053	000001F3	DATA	…		
11:15:54.894072	000001F3	CALL	QJOSNDJE	13	NO INTERRUPTS
11:15:54.894118	000001F3	RETURN	QSYSAUDR	12	
11:15:54.894120	000001F3	RETURN	QYPESVAC	11	
11:15:54.894122	000001F3	RETURN	QYPESVAC QYPEACAC AcGenerateAuditRecords	10	
			…		
11:15:54.894252	000001F3	CALL	**QMHSNDPM**	7	
11:15:54.899390	000001F3	CALL	QMHSNSTQ	8	
11:15:54.899878	000001F3	DATA	MESSAGE HANDLER SEND MESSAGE		
		DATA	MESSAGE ID -CPC3921		
		DATA	PROGRAM -QHST		

- First it checks if Service Tools Usage is audited
- The answer is “Yes”
- Then we prepare the data to be send to the journal entry, in the format for entry type ST = Use of Service Tools
- QSYSAUDR is the central API to log an audit journal entry. It receives most of the data for the audit journal entry from the programs that calls QSYSAUDR
- Data: ST => “Use of a Service Tool”, A => the only subtype for ST entries; “TS” => what service tool was used: STRTRC
- QSYSAUDR then uses the general QJSDNJE API to create a journal entry. That API is used for all journal entry writing, not just for audit journals.
- Note thread ID stays identical: Single threaded – “This job must focus on one thing, don’t do anything else before this has been logged”
- Interrupts are blocked during critical action—write of journal entry must be all-or-nothing, “atomic”
- Only *after* audit entry is logged can other action be taken, e.g. send a message to say trace has started
- Very fast: Total time used on audit-related things is about one tenth of a millisecond on a POWER8 server
- Sending an entry to the audit journal and a message to the history log is typical of

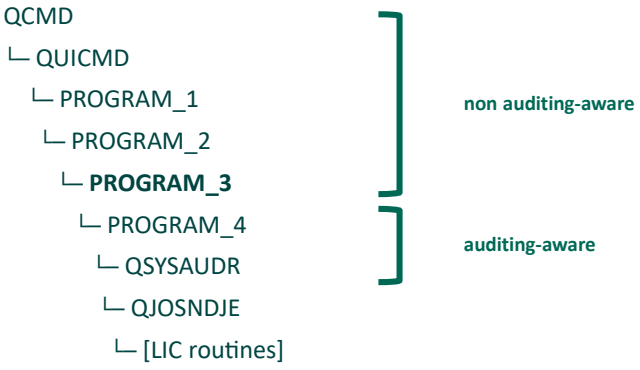
a mixed, “organically grown” auditing approach

- The difference is: A QHST message is a notification to the operator or admin; an audit journal entry is a forensic data point

I Digress: Which Program "Logs" the Entry?

How IBM i Logs Entries

- In most cases, the journal entry shows the **last program in the call stack that is not auditing-related** as "the" program that caused the entry
- Example:



FORTRA

I Digress: Which Program "Causes" the Entry?

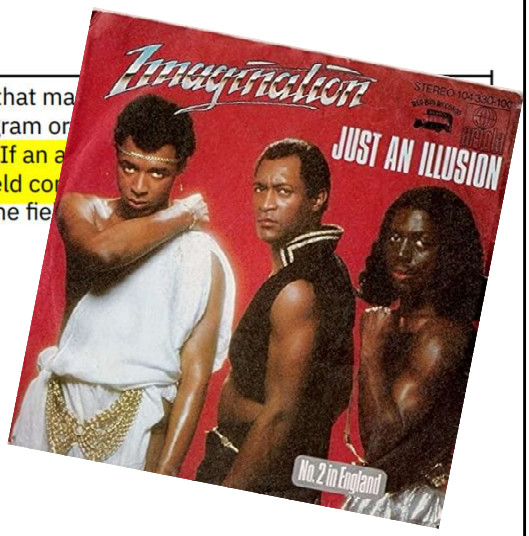
How IBM i Logs Entries

FORTRA

I Digress: Which Program "Causes" the Entry?

How IBM i Logs Entries

81	Program Name	Char(10)	The name of the program that ma the name of a service program or a compiled Java program. If an a not cause the entry, the field co program such as QCMD. The fie



39

FORTRA

QCMD is just a placeholder
What probably happened:

Starting the trace …
… called something in the *Licensed Internal Code* …
… that then "called back" to IBM i and triggered the log entry.

Which means the call stack shown was an illusion
This is how it can go if you have multiple layers of operating system

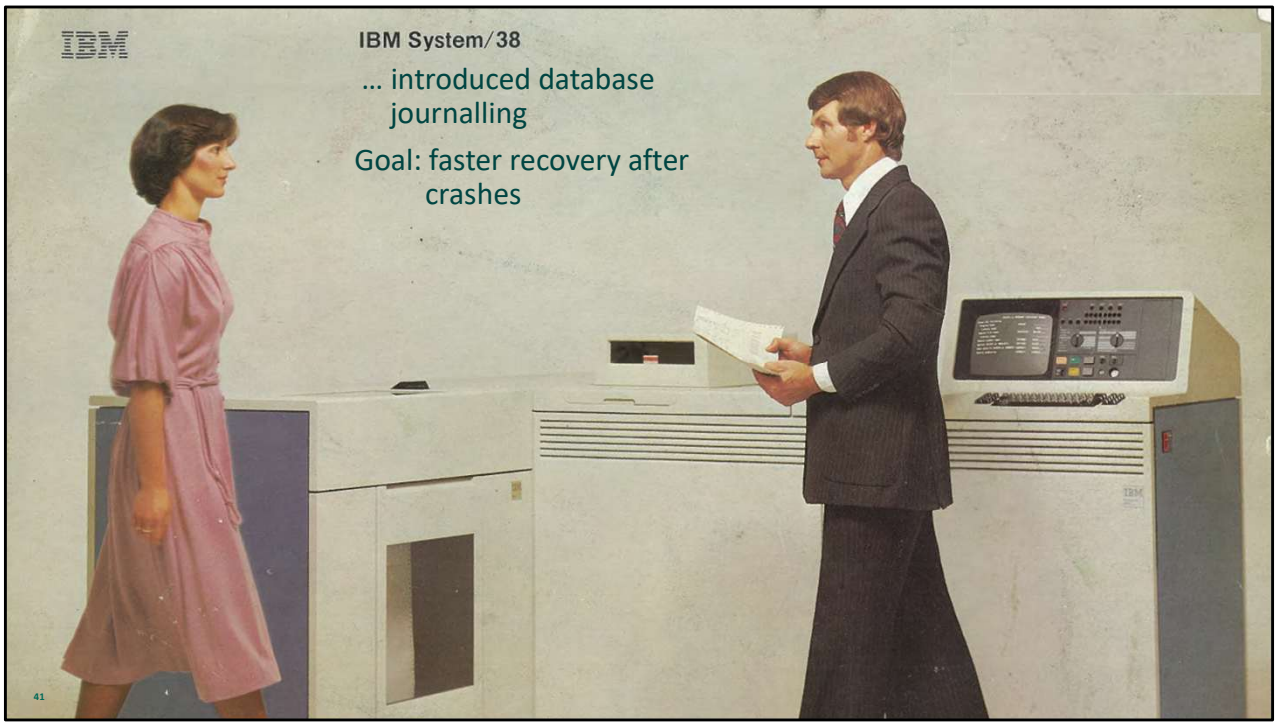
But now for something completely different:

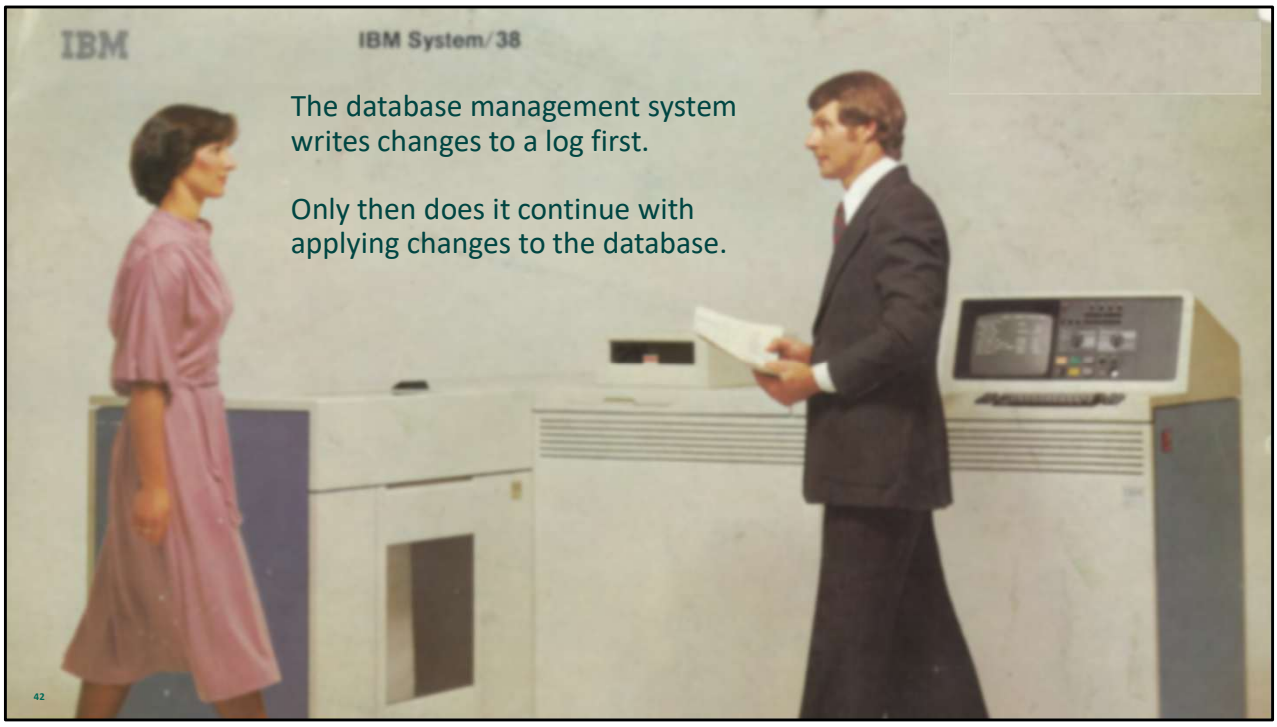
Why a Journal

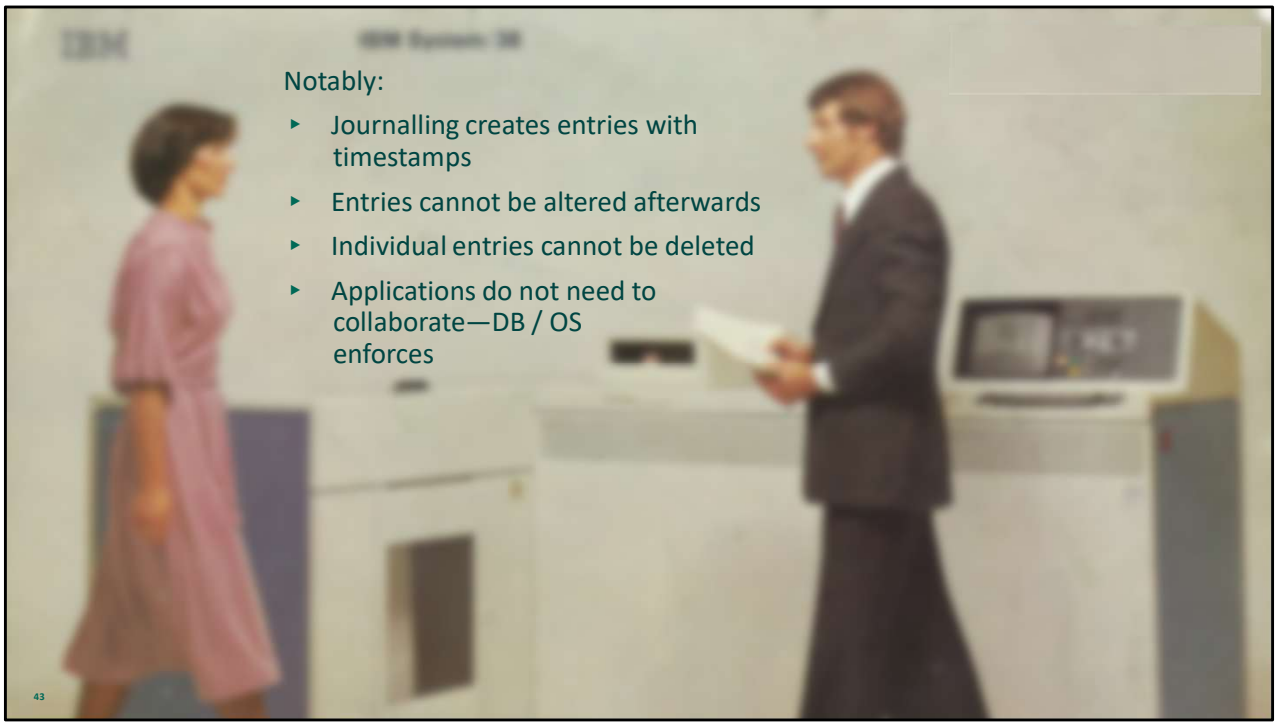
How IBM i logs Security Events

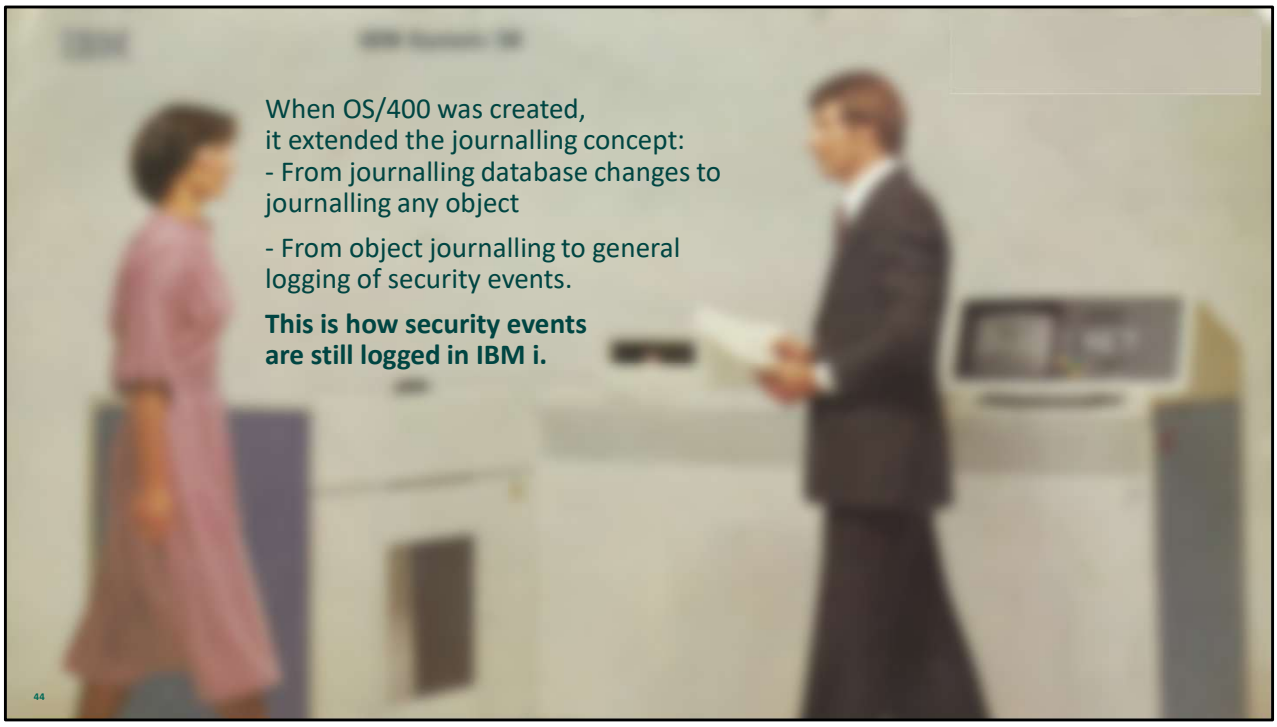
- Why a journal and not a file?
- … or some other internal-only object type?
- Let's think back …. way back …. to the 1970s

FORTRA


IBM
IBM System/38
… introduced database journalling
Goal: faster recovery after crashes
41


IBM
IBM System/38
The database management system writes changes to a log first.
Only then does it continue with applying changes to the database.
42


Notably:
Journalling creates entries with timestamps
Entries cannot be altered afterwards
Individual entries cannot be deleted
Applications do not need to collaborate—DB / OS enforces
43



-Security auditing uses a journal

-Tamper-proof log

-OS-enforced log writing

Fixed and Variable

How IBM i Logs Security Events

- Journal entries consist of:
  - A “fixed-length” header
  - A variable-length data section

* If toggled off, info is left blank in entry.

FORTRA

The Header

How IBM i Logs Security Events

- Fixed-length header entries are used for both “normal” journalling and security auditing
- Length of individual entries is fixed
- Roughly: Fixed = “independent of the type of journal entry and the specific event or file change”
- Some entries can are optional and can be toggled at journal level

* If toggled off, info is left blank in entry.

FORTRA

The Header

How IBM i Logs Security Events

- Actually fixed portion of the header—always logged, includes:
  - Journal code (T, U, …) ← always “T” for security events logged by the OS
  - Journal entry type
  - Timestamp
  - Sequence number (unique per receiver)…

* If toggled off, info is left blank in entry.

FORTRA

The Header: Optional Fields

How IBM i Logs Security Events

- Optional fields in fixed-length header that are written or not, depending on journal options:
  - Fully qualified job name
  - Effective user (current user) at time of event
  - Program that caused the event
  - Library + ASP device of the program
  - System sequence number, unique across all entries in all receivers on a system—relative ordering
  - Remote IP address
  - Thread identifier
- Why toggle them off at all?
  - To save disk space
- Do we want to toggle them off?
  - No!

The Data Section

How IBM i Logs Security Events

- The data is broken down into fields of different lengths
- The first field is always one byte long and represents the subtype of the logged event
- The remaining contents of the data section depend on the type of security event that was logged
- That is, on the **entry type** of the journal entry
- An AF entry has a different structure from a DO entry from a PW entry etc.
- All entry types have one or multiple subtypes

Table 185. JS (Job Change) journal entries (continued). QASYJSJE/J4/J5 Field Description File

Offset			Field	Format	Description
JE	J4	J5			
158	226	612	Job Subtype	Char(1)	The subtype of the job. ' ' No subtype D Batch immediate E Procedure start request J Prestart P Print device driver Q Query T MRT U Alternate spool user
159	227	613	Job Name	Char(10)	The first part of the qualified job name being operated on
169	237	623	Job User Name	Char(10)	The second part of the qualified job name being operated on
179	247	633	Job Number	Char(6)	The third part of the qualified job name being operated on
185	253	639	Device Name	Char(10)	The name of the device
195	263	649	Effective User Profile[2]	Char(10)	The name of the effective user profile for the thread
205	273	659	Job Description Name	Char(10)	The name of the job description for the job
215	283	669	Job Description Library	Char(10)	The name of the library for the job description
225	293	679	Job Queue Name	Char(10)	The name of the job queue for the job
235	303	689	Job Queue Library	Char(10)	The name of the library for the job queue
245	313	699	Output Queue Name	Char(10)	The name of the output queue for the job
255	323	709	Output Queue Library	Char(10)	The name of the library for the output queue

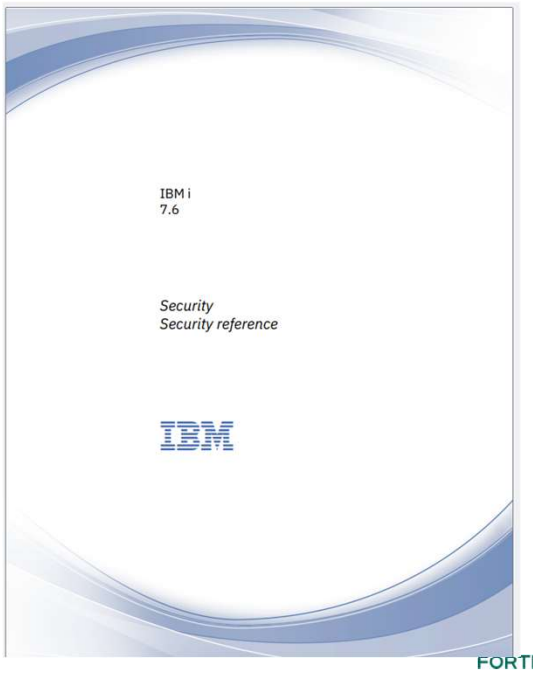
FORTRA

Where is the point in having an entry type with only one subtype? Consistency. It makers parsing the data easier.

Data Section: Where to Find Info

How IBM i Logs Security Events

- In the IBM i Security Reference for your respective IBM i version
- Appendix F

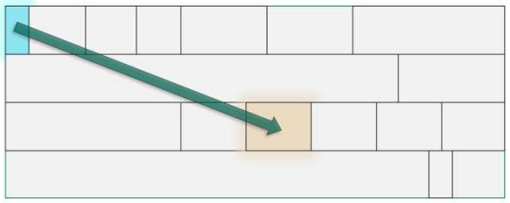


The Security Reference, not to be confused with the IBM i Security Guide

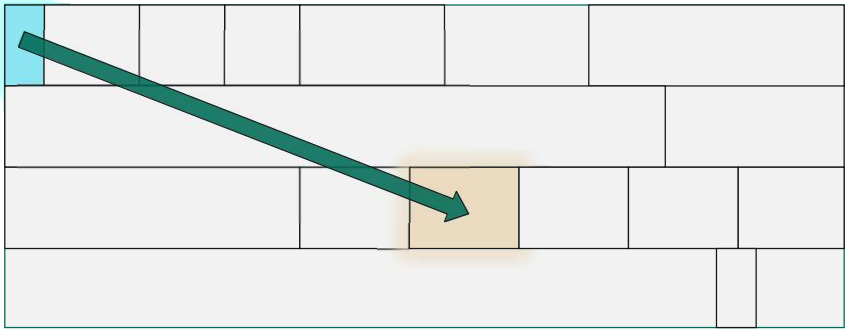
The Data Section: Dependencies

How IBM i Logs Security Events

- Sometimes, the *semantics* of a field depend on the value of another field:
  - If field A has value 'X' then field B represents <something>, else it represents <something else> or <is left blank>
- Sometimes, the *length* of a field is provided by another field in the same entry
  - "The following field is 956 bytes long"
- Dependencies are always within the same entry, never across entries



FORTRA



FORTRA

Changes Between Versions

How IBM i Logs Security Events

- Entry types, subtypes and other aspects of the audit journal entries' structure can change between OS releases
- The majority of these will stay the same, but you can expect a change of between 5 and 10 percent on new OS versions
- There are two kinds of reasons for these changes
  - To cover new functionality in the OS
  - To expand auditing to cover already existing functionality that was not in audit journal

FORTRA

- Usually no or only minimal changes in *Tech Refreshes or group PTFs*
- If you want to look for changes to the auditing structure, such as new entry types, get the IBM i Sec. Reference for the new release, then look at:
  - "What's New" section
  - Change indicators (black bars) in Appendix F

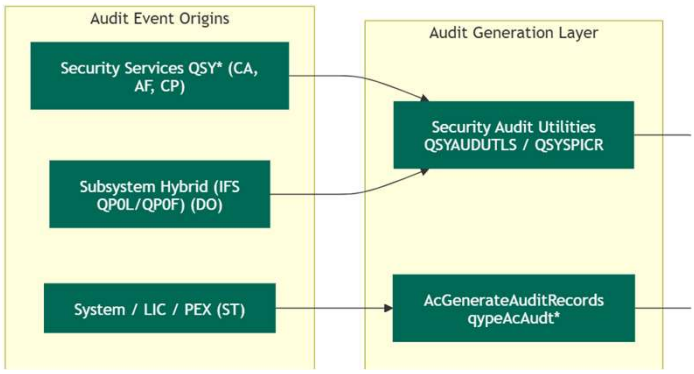
Changes Between Versions: Examples

How IBM i Logs Security Events

OS Version	Entry Type	Change	Type of Change
5.2	Authority Failure	New subtype K for special-authority violations—previously QHST messages	Cover existing functionality
7.2	RCAC Config	Auditing for new RCAC functionality in Db2	Cover new functionality
7.2	Change to User Profile	Adds fields for special authorities before a change	Cover existing functionality
7.4	Db2 Mirror	Auditing for new Db2 Mirror functionality	Cover new functionality
7.6	Authentication failure	New subtypes for built-in MFA, authentication exit point	Cover new functionality

Shared Infrastructure: Data preparation

Audit Journal Entry Creation



Different APIs can use the same infrastructure to format the data for the journal entry

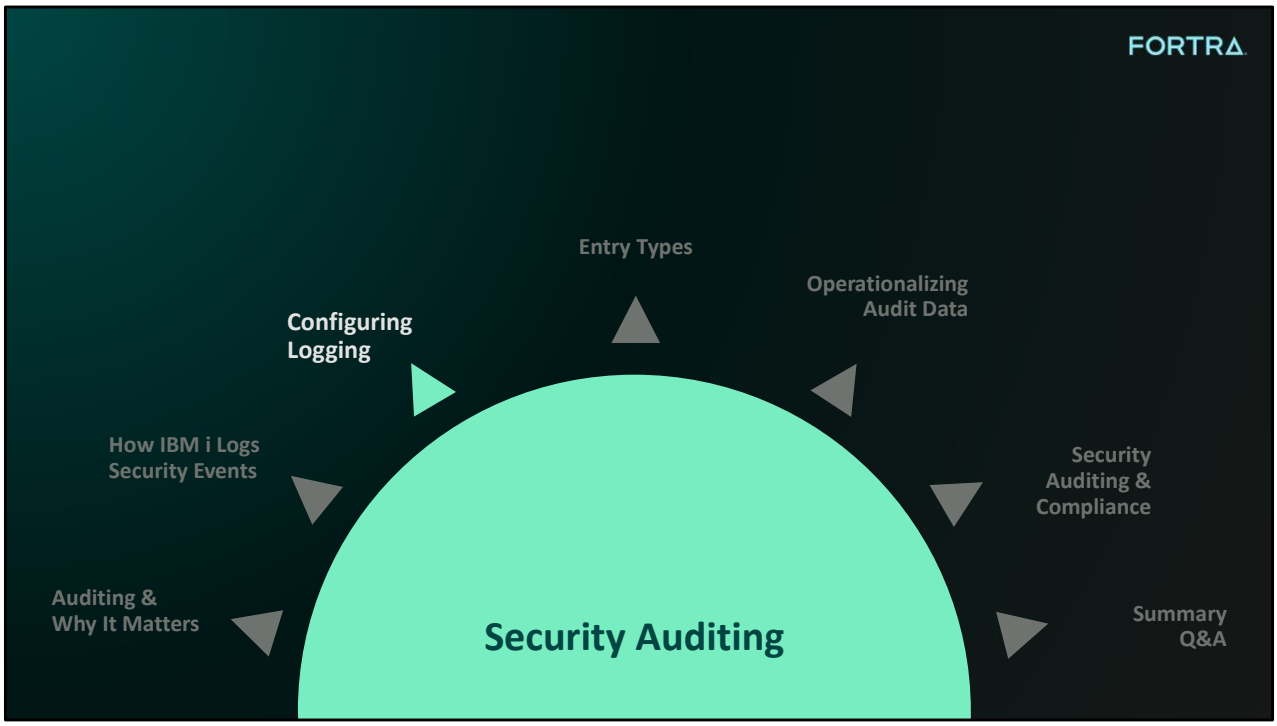
FORTRA

Summary: How IBM i Logs Security Events

- The security audit journal
- Where do events come from?
- Creation sequence
- Case study
- Why a journal
- The header
- The data section
- Changes between versions

FORTRA

…
So—do you think that that was a lot?
Because the next section is going to be even longer!


FORTRA
Entry Types
Operationalizing Audit Data
Configuring Logging
How IBM i Logs Security Events
Security Auditing & Compliance
Auditing & Why It Matters
Summary Q&A
Security Auditing

Configuring Logging

- Basics
- Audit journal and journal receivers
- System-wide auditing
- User action auditing
- Object access auditing
- Command auditing
- System values
- Tools
- Anti-Sabotage

Basics

Configuring Logging

- Auditing (logging) is not turned on by default
  - Maybe because auditing uses disk space
- You have to take action to turn it on
  - Create journal and receivers
  - Configure the logging settings
- There are three main kinds of logging settings:
  - System-wide
  - User profile action
  - Object access

Audit Journal and Receivers

Configuring Logging

- Journal
  - The audit journal has a hardwired name and library. It is always: Journal QAUDJRN in library QSYS
  - While security auditing is active:
    - System job QSYSARB2 always holds a lock on it
    - Trying to delete QAUDJRN fails, brings up error msg
  - As with every journal, the actual data is stored in journal **receivers**

Audit Journal and Receivers

Configuring Logging

- Receivers
  - Library: Journal receivers for QAUDJRN can be in ANY library
    - IBM recommends to NOT place them into QSYS lib
  - Name: Can use any name prefix you want
    - Leave space at end for numbering
    - Example: AUDRCV*
  - Receiver management:
    - Recommended: Have system (or additional software) automatically manage the receivers for QAUDJRN

Audit Journal and Receivers: Recommendation

Configuring Logging

- Journal QSYS/QAUDJRN ...
  - … exists and is linked to a journal receiver
  - … is set to include relevant data in “fixed-length data” header (IP address, etc.)
  - … has *PUBLIC *EXCLUDE authority
  - … is linked to a journal receiver
  - … is configured to let the system manage the receivers

FORTRA

We won’t go into details of journal management here

Audit Journal and Receivers: Recommendation

Configuring Logging

- Journal receivers:
  - Custom library, used only for QAUDJRN receivers
  - Name like “AUDRCV*nnnn*”
  - *PUBLIC *EXCLUDE authority
- HA Considerations
  - If you are using logical replication and the target system is running:
    - Do not replicate journal or receivers
    - Target system should have its own audit journal

Doing it the Hard but Morally Rewarding Way

Configuring Logging

- Create initial journal receiver and library
  - `QSYS/CRTLIB LIB(AUDRCVLIB) TYPE(*PROD) TEXT('Library for journal receivers for audit journal') CRTAUT(*EXCLUDE) CRTOBJAUD(*NONE)`
  - `QSYS/CRTJRNRCV JRNRCV(AUDRCVLIB/AUDRCV0001) TEXT('Security audit journal receiver') AUT(*EXCLUDE)`
- Create the security audit journal
  - `QSYS/CRTJRN JRN(QSYS/QAUDJRN) JRNRCV(AUDRCVLIB/AUDRCV0001) DLTRCV(*NO) RCVSIZOPT(*SYSDFT) FIXLENDTA(*JOB *USR *PGM *PGMLIB *SYSSEQ *RMTADR *THD) TEXT('Security Audit Journal') AUT(*EXCLUDE)`

FORTRA

System Values: Overview

Configuring Logging

The most important settings for the audit journal are controlled through system values:

- **QAUDCTL**—the Big Switch
- **QAUDLVL** and **QAUDLVL2**—system-wide “what to log”
- **QAUDENDACN**—the paranoid one

System Value QAUDCTL “Auditing Control”

Configuring Logging

- Function: Turns security auditing on and off.
- Supports multiple “values” = settings
- Allowed values:
  - ***AUDLVL**: This is the **big switch** to enable system-wide security auditing
  - *OBJAUD: Enables object access auditing (explained below)
  - *NOQTEMP: Disables object auditing for objects in QTEMP libraries (default option)
- Recommended: Use this only after you have set up journal and receivers
- Changes effective immediately

FORTRA

System Value QAUDLVL “Auditing Level”

Configuring Logging

- Function: Toggles **system-wide** logging for the different kinds of security events—say, *authority errors*, or *changes to exit programs*.
- Toggles object access auditing in general (explained below)
- Introduced in V2R3
- Values: *AUTFAIL, *CREATE, *DELETE … Toggle for Individual types of events
- Support multiple “values” == settings, e.g. *CREATE + *DELETE + *SECURITY
- Sequence of values does not matter

Note: The names indicates a scalar “level” similar to password level or system security level, so that probably was the idea, but it actually can contain multiple settings.
In 1993 IBM introduced OS/400 V2R3. At that time, IBM was still trying to position the AS/400 as a system for the US military, so IBM sought a “Trusted System Evaluation Criteria” or “Common Criteria” C2 certification. To that end, it made changes to the OS, such as introducing the *AUDIT special authority and expanding auditing capabilities.

System Value QAUDLVL2 “Auditing Level Extension”

Configuring Logging

- QAUDLVL is the “classical” system value to configure.
- As new types of events kept getting added, QAUDLVL ran out of space
- IBM added the QAUDLVL2 “extension” system value.
- If QAUDLVL contains *AUDLVL2 ➔ values in QAUDLVL2 are used to configure system-wide auditing
- *AUTFAIL, *CREATE, *DELETE … Same as for QAUDLVL

FORTRA

System Values QAUDLVL + QAUDLVL2

Configuring Logging

- For both QAUDLVL and QAUDLVL2, changes affects logging instantaneously—no need to restart auditing or IPL!
- Recommendation:
  - QAUDLVL: Only *AUDLVL2
  - See end of this section for QAUDLVL2 recommendations

System Value QAUDFRCLVL “Force Auditing Data”

Configuring Logging

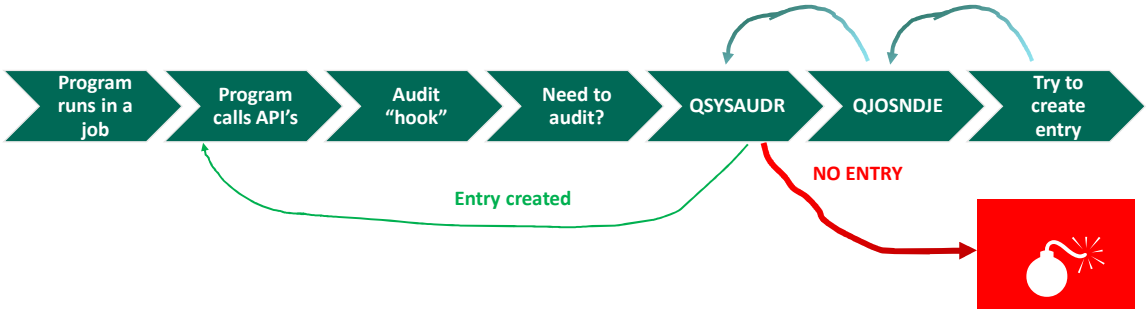
- Function: Controls how frequently audit journal entries that have been “written” are flushed out to disk
- Settings:
  - *SYS Let IBM i decide ← **Use this!**
  - 1—100 Don’t bother

FORTRA

System Value QAUDENDACN “Auditing End Action”

Configuring Logging

- Function: Controls what happens if audit journal entry CANNOT be written
  - E.g.: QAUDJRN does not exist; receiver damage is preventing a write
  - Not a normal situation!
- Settings:
  - *NOTIFY Lets user know and stop writing audit journal entries ← Default, **Use this!**
  - *PWRDWNSYS What it says!! ← For the truly paranoid only



71

FORTRA

Of course the actual check happens in QSYSAUDR, QJOSNDJE does not care about QAUDENDCN

*PWRDWNSYS will also turn of security auditing in QAUDCTL, otherwise you would never be able to get back into your system

System Value QAUDJRL

Configuring Logging

- Function: Like QAUDCTL and QAUDLVL combined
- OLD, got dropped from the OS around 1994
- If you have used this: Take it easy, my friend!

FORTRA

How to Think About Tools

Configuring Logging

- With “tools”, I mean API’s, CL commands, SQL functions and procedures
- You will get to know a useful “coordinate” system for audit journal tools of any kind
- It’s 4-dimensional!
- You can position any tool—API, CL command, SQL function…—along those 4 dimensions, and that already tells you most of what you need to know *
- The rest you can read up in IBM documentation

* We won’t discuss API’s in much detail here.

How to Think About Tools

Configuring Logging

- The coordinates/dimensions:

<Name of tool>	
Dimension	Values
Object/s	Examples: Journal; journal receivers; system value
Interface Type	API, CL command, SQL function/view/procedure, GUI
Object Access	Read Y/N, Write Y/N
Auditing-Aware?	Auditing-aware *or* Not auditing-aware

Read/Display/Retrieve Yes or No, and Write/Change/Set/Create Yes or No
The non-auditing specific tools for *reading* can be used to read out auditing-aware information, but may need additional interpretation, for instance for parsing audit journal data.
The non-auditing-aware tools for *writing* can be used to affect various settings, including the auditing related ones.

How to Think About Tools: Example

Configuring Logging

CHGSECAUD "Change Security Auditing" command	
Axis	Examples
Object/s	QAUDJRN journal Journal receivers for QAUDJRN Library for receivers QAUD* system values
Interface Type	CL command
Object Access	Write
Auditing-Aware?	Auditing-aware

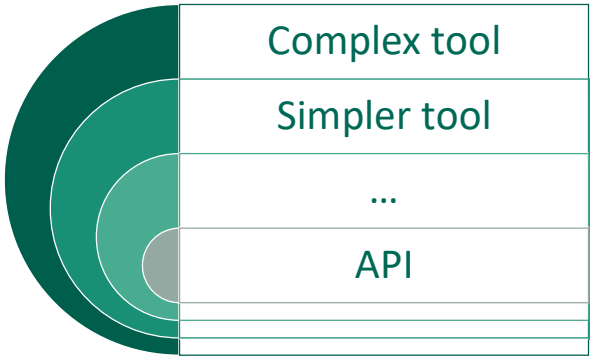
FORTRA

CHGSECAUD is actually both a Write and a Read tool: When you prompt the command, you can see the current settings for the system values etc. But its primary function is to write.

How to Think About Tools: Tools Relations

Configuring Logging

- Complex tools are built on simpler tools
- Examples:
  - Navigator GUI
    - Uses SQL views/functions/procedures
      - … That use
        - API's and
        - CL commands … which use APIs



FORTRA

Tools for System Values: CL Commands

Configuring Logging

Type	Reading—auditing-aware	Reading—non auditing-aware	Setting—auditing-aware	Setting—non auditing-aware
CL	DSPSECAUD PRTSYSSECA*	DSPSYSVAL	CHGSECAUD CFGSYSSEC**	CHGSYSVAL

* Output includes other system values as well

** Makes other changes as well. Only use if you really know what you are doing

FORTRA

Tools for System Values: SQL

Configuring Logging

Type	Reading—auditing-aware	Reading—non auditing-aware	Changing—auditing-aware	Changing—non auditing-aware
SQL	QSYS2.SECURITY_INFO: AUDIT_JOURNAL_EXISTS AUDIT_JOURNAL_RECEIVER_LIBRARY AUDIT_JOURNAL_RECEIVER AUDITING_CONTROL AUDITING_LEVEL AUDITING_LEVEL_EXTENSION	QSYS2.SYSTEM_VALUE_INFO	-/-	(QSYS2.QCMDEXC + CL commands)

78

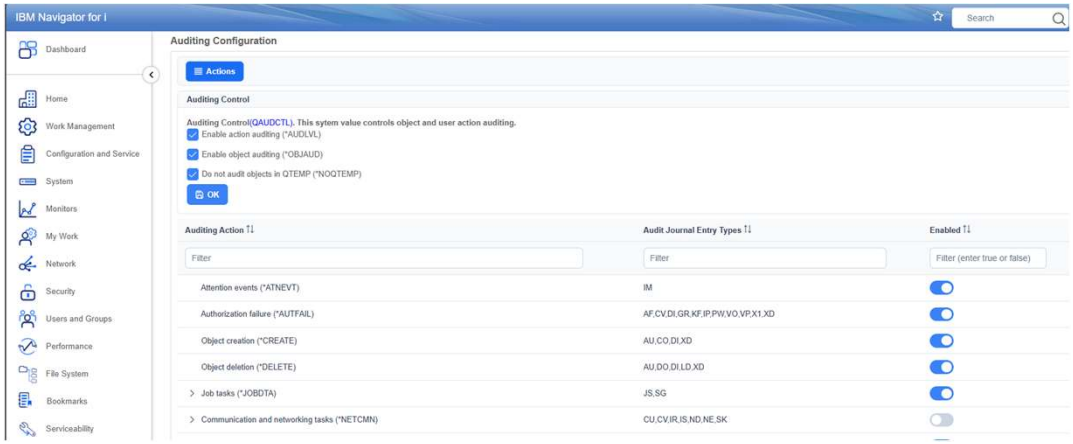
FORTRA

The QSYS2.SECURITY_INFO view is the SQL counterpart to the DSPSYSAUD command
SQL ones are all read-only—looks like a conscious design decision by IBM—they are probably extra careful about allowing write access through SQL
If you wanted to **set** values “with SQL”, there is no dedicated SQL procedure. You could only use QCMDEXC/QCMDEXC() + CL commands

now let’s take a quick look at how you would view or define audit settings in the GUI

Doing it the GUI Way

Configuring Logging



Navigator for i: **Manage system → Security → Audit Journal → Auditing Configuration**

FORTRA

This is the Navigator counterpart to the CHGSECAUD command

System-Wide, Object and User Logging

Configuring Logging

- So far, we have looked mainly at system-wide log configuration
- That kind of configuration means events will be logged based only on the type of event
- There are also **object access logging** and **user action logging**
- These additionally require:
  - configuration of a user profile or object and
  - involvement of that user or object in the event

  for the event to be logged.

	System-Wide	Object Access	User Action
Type of action	✓	✓	✓
Specific object		✓	
Specific user profile			✓

FORTRA

"Involvement" for user profile means: the profile is the actor
for object means: the object is affected,—ead or changed

Object Access Logging

Configuring Logging

- Creates an audit entry when an object is *read* or *changed*
- Introduced in V2R3
- **R**eads create Z**R** audit entries
- **C**hanges create Z**C** audit entries
  - ZC entries do not contain details of the change, they only record the fact *that* an object was changed

FORTRA

Why the 'Z', you wonder? Because the other letters were already taken …
ZC: It's not DB journalling! You don't get before- and after-images

"Reading" and "changing" are mapped to object type-specific actions
E.g.: Running a program = Read action
See Security Reference *Appendix D* for mappings

Object Access Logging: Setup

Configuring Logging

- Setup from object:
  - Set “object auditing value”
- Setup from user profile:
  - Set “object auditing value”
- This is how the two settings interact to determine logging:

User \ Object	Object *NONE	Object *CHANGE	Object *ALL	Object *USRPRF
User *NONE	(None)	Change Access	All Access	(None)
User *CHANGE	(None)	Change Access	All Access	Change Access
User *ALL	(None)	Change Access	All Access	All Access

When the Object’s setting is *NONE, *CHANGE, or *ALL, that value is applied. (top of column is applied)
Only when the object’s setting is *USRPRF, then the user profile’s setting is applied. (left of row is applied)

Object Access Logging: Tools

Configuring Logging

Type	Reading—auditing-aware	Reading—non auditing-aware	Changing—auditing-aware	Changing—non auditing-aware
CL	-/-	DSPOBJD (native), DSPATR (IFS) DSPUSRPRF	CHGOBJAUD (native), CHGAUD (IFS) CHGUSRAUD	-/-
SQL	-/-	QSYS2.OBJECT_STATISTICS function QSYS2.USER_INFO_BASIC view	-/-	(QSYS2.QCMDEXC)

The QSYS2.SECURITY_INFO view is the SQL counterpart to the DSPSYSAUD command

SQL ones are all read-only—looks like a conscious design decision by IBM—they are probably extra careful about allowing write access through SQL

If you wanted to **set** values “with SQL”, you could only use QCMDEXC/QCMDEXC() + CL commands

User Action Logging

Configuring Logging

- Override per user profile: which types of events should be logged
- Think: “Profile-level setting in addition to QAUDLVL/2”
- Example:
  - You want to track spool file operations only for one specific user
  - Setting name for spool file operations is *SPLFDTA
    - ➔ From QAUDLVL2: Omit *SPLFDTA
    - ➔ For specific profile, set profile’s “user action auditing” to *SPLFDTA

FORTRA

User Action Logging

Configuring Logging

- Introduced in V2R3
- Profile settings only *add to*, never *subtract* from, system-level setting
- Tools: Same as for object access
- Settings' names are same as for system-level, except:
- Special value: *CMD

FORTRA

- Special value *CMD leads us to …

Command Logging

Configuring Logging

- Command logging aka command auditing **logs the execution of CL commands**
- It can be achieved through *object access logging* or *user action logging*
- Both kinds create the same kind of **CD** (Command Usage) audit journal entries when CL commands are run
- CD entry contains the library and name of the executed command, command parameters, and other info

FORTRA

The content and structure of the CD entries is identical regardless of which configuration caused the entry

Command Logging: Setup

Configuring Logging

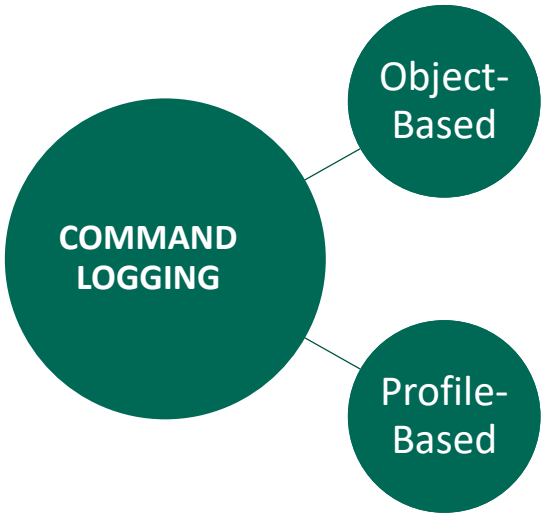
- Object-based setup:
  - Object audit for the *CMD object is set to *ALL
    and
  - QAUDCTL contains the value *OBJAUD
- User profile-based setup:
  - User action auditing for the user profile includes *CMD
    and
  - QAUDCTL contains the value *AUDLVL

FORTRA

Object- or command-based setup
Note that both lead to a CD entry just under different conditions.
Also note, that normally, running the command is classified as a “Read” operation and would create a ZR = “Object Read” entry, but things work differently for command objects.

Command Logging: Command- vs. User-Driven

Configuring Logging



- "Always log that command when it is run"
- Use for commands with high impact
- E.g.: PWRDWNSYS, ENDSBS

- "Log all commands run by that user"
- Track a user's actions
- Use for profiles with special authorities

FORTRA

Command Logging: Command- vs. User-Based

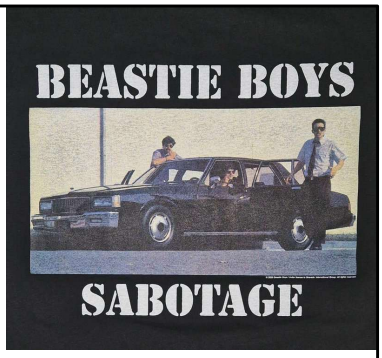
Configuring Logging

- User profile-based setup will create a lot of “noise” in entries
- Think: ADDLIBLE, CHKOBJ, ALCOBJ, DLCOBJ commands
- Can’t be avoided!
- Which one should you use, object- or user-based?
  - Both!

Anti-Sabotage

Configuring Logging

- Bad things someone could do to hide their tracks
  - Turn off system-wide auditing
  - Turn off user action logging
  - Turn off object access logging
  - Copy a powerful user profile and perform actions under a different user
- What can you do about it?
  - Preventative controls
  - Detective controls



FORTRA

preventative = active, preventing bad things or reducing the likelihood that they happen
detective = create an audit trail

Anti-Sabotage: Preventative Controls

Configuring Logging

Risk	Defender Action
Someone could turn off system-wide auditing	1) Don't hand out special authorities like candy
Someone could turn off user action auditing	
Someone could turn off object access auditing	2) Limit access to QSECOFR
Someone could copy powerful profile etc.	

FORTRA

Anti-Sabotage: Detective Controls

Configuring Logging

Risk	Defender Action
Someone could turn off system-wide auditing	QAUDLVL2: Include *SECCFG
Someone could turn off user action auditing	
Someone could turn off object access auditing	
Someone could copy powerful profile etc.	

FORTRA

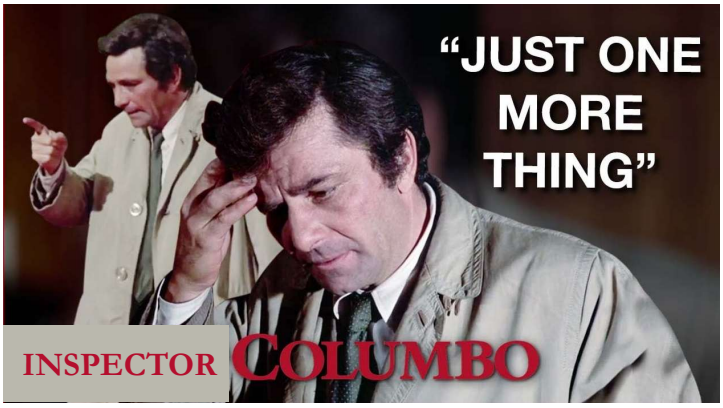
Track SV entries
Track AD entries
Track AD entries
Track CP entries

Anti-Sabotage: One More Thing

Configuring Logging

What if someone did not just disable auditing (changing the future), but…

… deleted the QAUDJRN journal receivers (erasing the past)?



93

FORTRA

We will get back to this in our next section, "Operationalizing Audit Data".

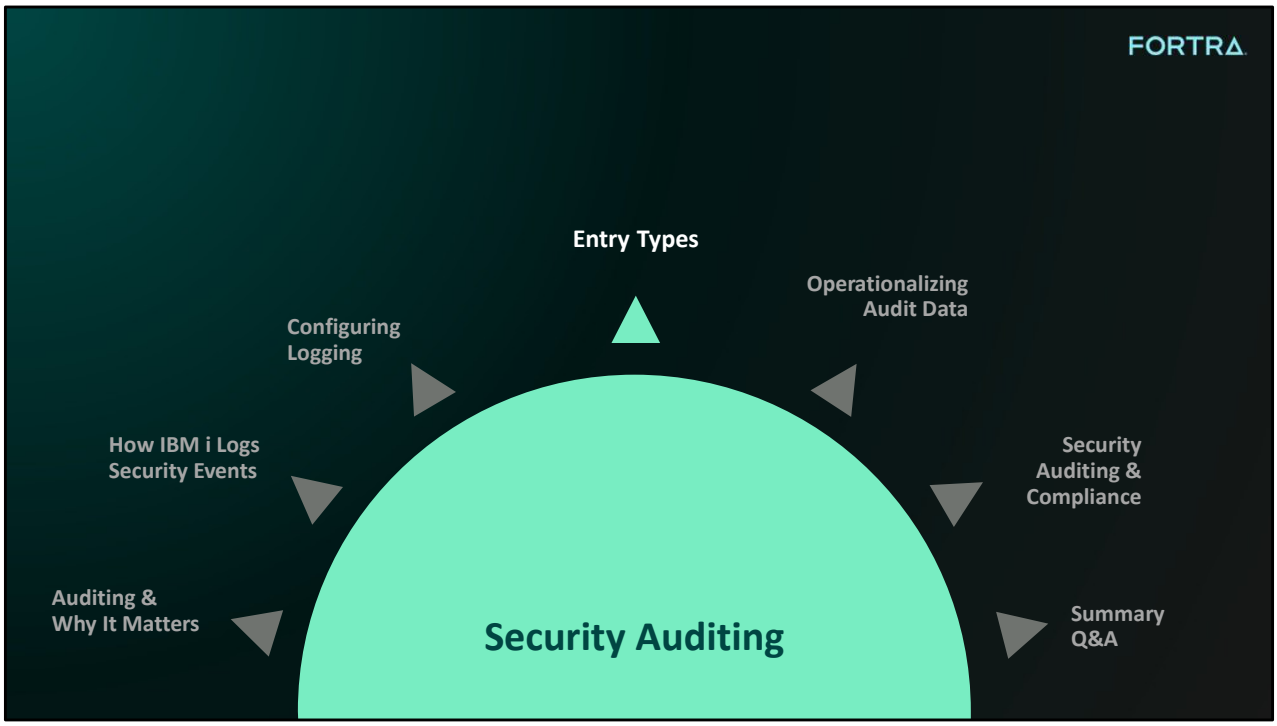
Section Summary

Configuring Logging

- That was a lot!
- Basics
- Audit journal and journal receivers
- System-wide auditing
- User action auditing
- Object access auditing
- Command auditing
- System values
- How to think about tools
- Anti-Sabotage

FORTRA

Note: In ST = Service Tool Usage entries, you sometimes see that a DMPOBJ was run. That is usually the OS collecting info after a program crashed. You can ignore that.


FORTRA
Entry Types
Operationalizing Audit Data
Configuring Logging
How IBM i Logs Security Events
Security Auditing & Compliance
Auditing & Why It Matters
Summary Q&A
Security Auditing

Entry Types

- Basics
- Very useful entry types
- Rarely useful
- Things that ain’t where you’d expect
- Missing in action

FORTRA

Basics

Entry Types

- After *journal codes*, entry types provide the most general indication of the content of a journal entry
- The entry type represents the high-level type of security event that is logged
- It is represented a two-digit ID
- This ID can be mapped to the human-readable entry type name
- The ID is often an acronym of the human-readable name, e.g., “CD” ← “**C**omman**d** string”

FORTRA

Very Useful Entry Types

Entry Types

Number	Type ID	What It's For
10	**SK**	See your incoming TCP and UDP connections!
9	**GR**	Changes to exit points, exit programs, function availability blocking functionality
8	**JS**	Jobs started, ended, submitted, user changed, ….
7	**SV**	Change of system values
6	**DO**	Go here to see when profiles are deleted
5	**CP**	User profiles created, disabled, otherwise changed
4	**AF**	Authority Failures—no object authority, no special authority, not allowed in QSECURITY 50, or scanning found file to be infected with a virus
3	**CD**	CL commands executed
2	**VP**	"Network password" error = failed NetServer authentication
1	**PW**	Failed authentication, except for NetServer

98

FORTRA

CP: \\except profile description!
GR-"generic record"—I mean, come on, IBM!!
JS – "Job stuff"
AF: *PGMFAIL, *AUTHFAIL settings in QAUDLVL/2

Rarely useful: IP (Interprocess Communication)

Entry Types

- "This job has interacted with that job… they exchanged information …. It is SUSPICIOUS!"
- "And they do it without an object LIKE ANY GOD-FEARING JOB SHOULD! It is WRONG!"
- Many ported applications use interprocess comms extensively create lots of events
- Avoid logging it: Instead of using *SECURITY, use all the *SEC*xxxx* QAUDLVL/2 values *except for* *SECIPC
- Only real use scenario: You are paranoid

Things That Ain’t Where You Would Expect

Entry Types

- Sign-on and sign-off events?
  - No entry type!
  - Workarounds:
  - T:JS start, end, and disconnect events for job type = Interactive
  - T:JS “job modified” events for job name = QZDASOINIT, QTFTP*, … where current user is set
  - Exit points (sign-on server)

FORTRA

Things That Ain't Where You Would Expect

Entry Types

- User profile deleted
  - Profile created? → CP!
  - Profile changed? → CP!
  - Profile deleted ….. T:DO "Object Deletion" !!??!

FORTRA

Missing in Action

Entry Types

- FTP Activity
- SFTP activity
- SSH sign-on
- SST configuration
- Changes to job scheduler entries

102

FORTRA

FTP activity

GETs, PUTs

Have to use exit points to get this

SSH / SFTP / SCP sign-on

No event

“No such number, no such phone” —Elvis

“We will port this weird Unix stuff, but we don’t like it and we will not honor it with an entry type”

Also no exit points!

Workaround: Turn on syslog for SSH and SFTP

SST configuration

You can see someone started SST…. But not what they configured in there!

Workaround: You have to use the SST’s/DST’s own log for that

Weird exception: C3 = Changes made IN ONE SPECIFIC tool in SST do get logged!

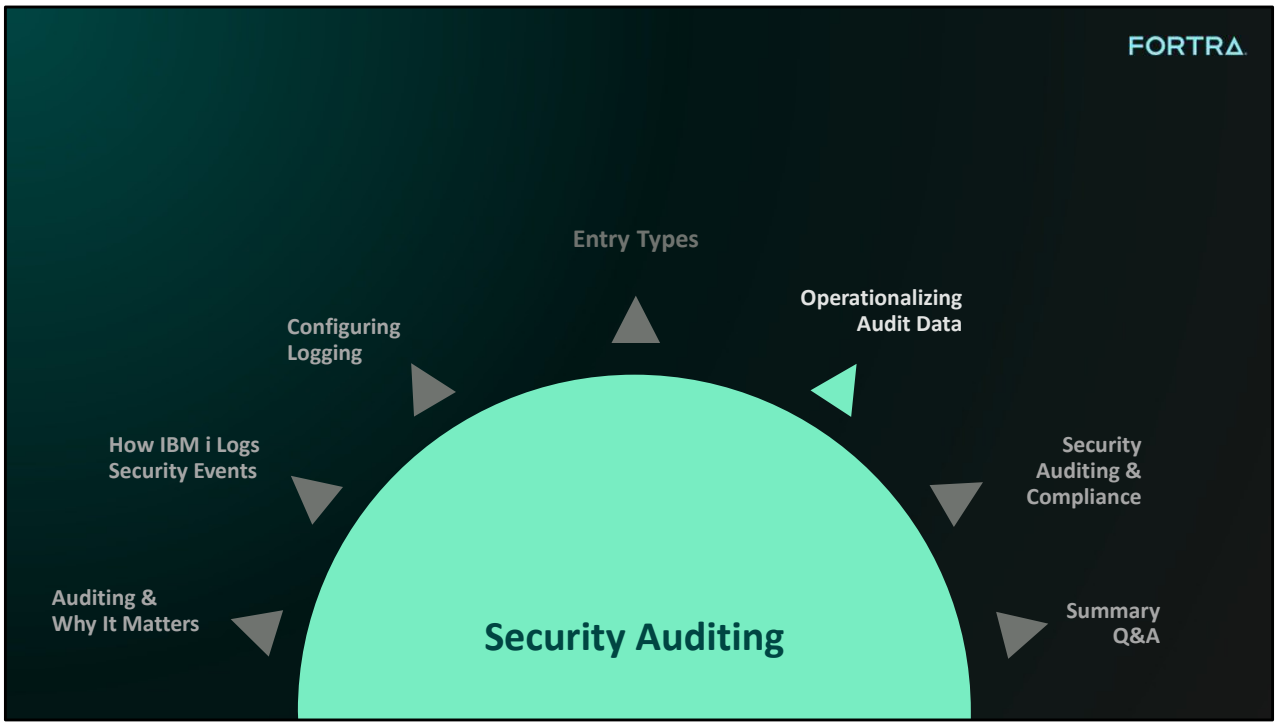
Changes to job scheduler entries

You can track when someone changed the scheduled entries… but not *what*

they changed
OK, admittedly, that’s more about system operations… But still!!
Workaround: Periodic WRKJOBSCE’s to *PRINT

Summary: Entry Types

- Basics
- Very useful entry types
- Rarely useful
- Things that ain’t where you’d expect
- Missing in action


FORTRA
Entry Types
Operationalizing Audit Data
Configuring Logging
How IBM i Logs Security Events
Security Auditing & Compliance
Auditing & Why It Matters
Summary Q&A
Security Auditing

Operationalizing Audit Data

- … AKA “reading out and using”
- Audit journal enabled?
- Use cases
- Reading audit journal data
- Analyzing data with SQL
- Seeking imbalance
- Applications

FORTRA

Audit Journal Enabled?

Security Auditing and Why It Matters

2026 State of IBM i Security Study:

93%

of all respondents had the security audit journal enabled (2017: 85%)

Audit Journal Enabled!

Security Auditing and Why It Matters

Great! So how do you use it?

Well, that depends on what you want to achieve.

The most important use cases are:

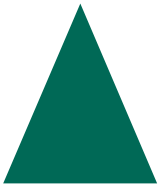
Main Use Cases

Security Auditing and Why It Matters

REGULAR REVIEW

Periodic review

Goal:
Pick up anything unusual



MANUAL INVESTIGATION

Specific investigation, based on review or known incident

Goals:
Confirm incident
Find root cause
Prepare mitigation

AUDIT

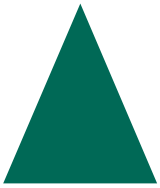
Document what was happening and that you are tracking events

Goal:
Make the auditor happy

FORTRA

Main Use Cases

Security Auditing and Why It Matters



SIEM/SOAR Integration

Goals:
Alerting
Starting incident process
Integrate IBM i
Cross-correlation, contribute to organizational situational awareness

FORTRA

Reading Audit Journal Data: Intro

Operationalizing Audit Data

- Types of data to retrieve data from the audit journal
- We will use similar “dimensions” as before:
  - Tool type: CL vs. SQL vs. GUI
  - Auditing-aware vs. non-auditing-aware, general-purpose journal tools

110

FORTRA

- DSPAUDJRNE is dated, but sometimes useful for quick checks in the “classical” entry types. IBM stopped updating it at some point before V5R4 and since then has not updated entry types, subtypes etc. for this command.
- CPYAUDJRNE in contrast is kept up-to-date. You can only output data for a single entry type at a time, because the outfile’s columns depend on the structure of the selected entry type

Reading Audit Journal Data: Auditing-Aware Tools

Operationalizing Audit Data

- Auditing-aware tools:
  - Can not just read the variable-length entry data, but parse the data into fields for on the different audit journal entry types
  - Can reflect that structure in their output, creating outfiles with columns corresponding to entry type-specific fields
  - Can **verbalize** the data so that humans can understand it, e.g. translate subtypes into text or verbalize an entry as “Someone tried to sign on with an unknown user profile from IP address 1.2.3.4”.

FORTRA

Note that you can also creste your own verbalization, you donÄr have to user that of any tool. However, tht means more work, e.g. to write an SQL query.

Reading Audit Journal Data: CL Commands

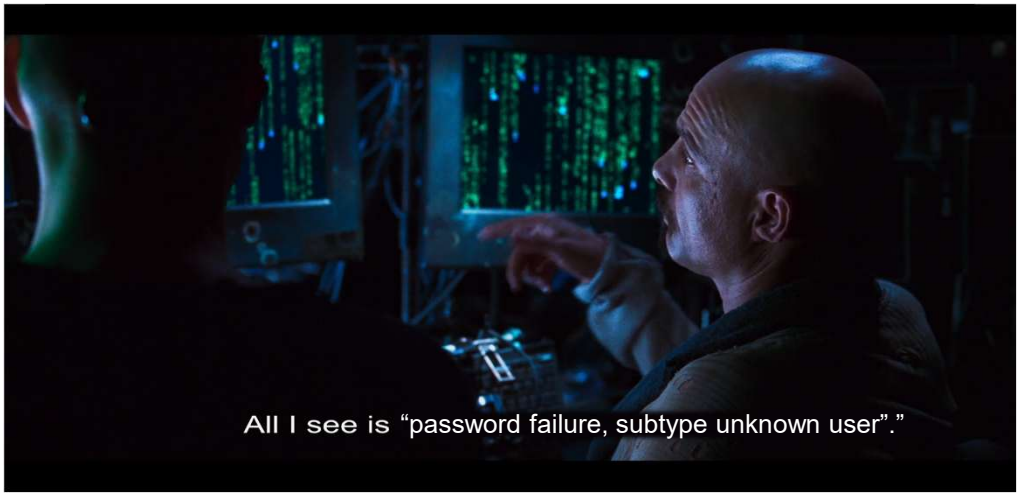
Operationalizing Audit Data

Type	Reading—auditing-aware	Reading—non-auditing-aware
CL	DSPAUDJRNE ← no longer updated	**DSPJRN** ← my "go-to" tool for quick checks
	CPYAUDJRNE ← "copy entries to a file"	RCVJRNE — for CL programs only
	QUSRTOOL/DSPAUDLOG (removed in V3R6)	

- DSPAUDJRNE is dated, but sometimes useful for quick checks in the "classical" entry types. IBM stopped updating it at some point before V5R4 and since then has not updated entry types, subtypes etc. for this command.
- CPYAUDJRNE in contrast is kept up-to-date. You can only output data for a single entry type at a time, because the outfile's columns depend on the structure of the selected entry type
- DSPJRN is a little bit auditing-aware – shows the entry types of the entries

Reading Audit Journal Data: DSPJRN Example

Operationalizing Audit Data



113

FORTRA

Here’s example output from DSPJRN. DSPJRN is actually a tiny bit of auditing-aware, it translates the entry type, e.g. a PW entry is shows as type “Invalid password or user ID “, but that’s all. The data section of the entry is not parsed or verbalized.

Reading Audit Journal Data: Tools

Operationalizing Audit Data

You can only use the CPYAUDJRNE command to output QAUDJRN entries for *one* entry type at a time.

What could be the reason?

Hint—this limitation does not exist if you use `DSPJRN QAUDJRN OUTPUT(*OUTFILE)`

FORTRA

The reason is that CPYAUDJRNE is an audit-aware tool that creates output file with columns that reflect the fields of the respective entry type. Since different entry types have different fields, the output files must have different columns, too. DSPJRN in contrast treats all data in the journal entry (variable-length part) as a single data blob and consequently does not impose any further structure on the outfile.

Reading Audit Journal Data: SQL

Operationalizing Audit Data

Type	Reading—partially-auditing-aware
SQL	`QSYS2.DISPLAY_JOURNAL()` function - Can verbalize audit journal entry - Cannot fully parse data into fields Helper function: `QSYS2.JOURNAL_CODE_INFO` view (7.6)

- DISPLAY_JOURNAL() This is the SQL counterpart to the DSPJRN command
- You can optionally use the JOURNAL_CODE_INFO view to translate the entry types, like “PW”, into something human-understandable.

Reading Audit Journal Data: SQL

Operationalizing Audit Data

Sample verbalization from DISPLAY_JOURNAL(), using the option to generate a Syslog entry:

```
<38>1 2026-06-13T08:12:10.417088Z MY_SYSTEM.FORTRA.COM - - - -
CEF:0|IBM|IBM i|7.4|QSYS-QAUDJRN|T-CD|Low|reason=Command string
audit msg=Command run from a compiled OPM CL program or an ILE
CL Program fileType=*CMD cs1Label=objName cs1=QSYS/ALCOBJ
suser=QSECOFR sproc=799397/QUSER/QZDASOINIT shost=MY_SYSTEM
src=10.1.2.3 spt=54526
```

116

FORTRA

- The option to generate a Syslog entry will create the content of a Syslog entry including the verbalization of the entry,
- here, the verbalization used multiple fields of the underlying CD entry, The subtype, the “Where run”, the command object name and library.

Reading Audit Journal Data: SQL Tools

Operationalizing Audit Data

Type	Reading—auditing-aware
SQL	`SYSTOOLS.AUDIT_JOURNAL_`*XX* table function `QSYS2.MANAGE_AUDIT_JOURNAL_DATA_MART` procedure

FORTRA

- These tools are auditing aware
- They will parse the journal entries into entry type-specific fields and provide verbalization

Reading Audit Journal Data: SQL

Operationalizing Audit Data

SYSTOOLS.AUDIT_JOURNAL_*XX* functions are SQL table function. The “XX” is replaced with the ID of desired entry type, e.g. “CD”, “AF” or “ST”.

Functions return the entries from the audit journal that match the entry type *XX* and any other specified criteria

```
SELECT VIOLATION_TYPE CONCAT ' - ' CONCAT VIOLATION_TYPE_DETAIL,
       COUNT(*) AS VIOLATION_COUNT
  FROM TABLE (
      SYSTOOLS.AUDIT_JOURNAL_PW ( )
    )
  GROUP BY VIOLATION_TYPE CONCAT ' - ' CONCAT VIOLATION_TYPE_DETAIL
  ORDER BY 2 DESC;
```


FORTRA

- It’s the SQL equivalent to CPYAUDJRNE

Reading Audit Journal Data: SQL

Operationalizing Audit Data

QSYS2.MANAGE_AUDIT_JOURNAL_DATA_MART is an SQL procedure that reads journal entries, and creates a table, or updates a table, with the journal entry data, or that deletes such a table.

The table's library name and the entry type are provided by the user. The table's SQL name and system name reflect the entry type.

The list of data mart tables can be queried with QSYS2. AUDIT_JOURNAL_DATA_MART_INFO.

The procedure uses the SYSTOOLS.AUDIT_JOURNAL_*XX* table functions to read and humanize the data.

```
CALL QSYS2.MANAGE_AUDIT_JOURNAL_DATA_MART(JOURNAL_ENTRY_TYPE => 'PW',
                     DATA_MART_LIBRARY => 'DMARTLIB',
                     STARTING_TIMESTAMP => CURRENT DATE - 1 MONTH,
                     ENDING_TIMESTAMP => CURRENT TIMESTAMP);
```

119

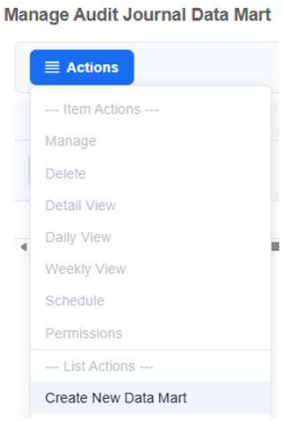
FORTRA

For instance, for PW entries, the SQL table name for PW entries is AUDIT_JOURNAL_PW, the system name for the file is AJ_PW.

The DATA_MART procedure is used for the Navigator for i GUI for reviewing audit journal entries. Let's see that in action!

Reading Audit Journal Data: Navigator for i

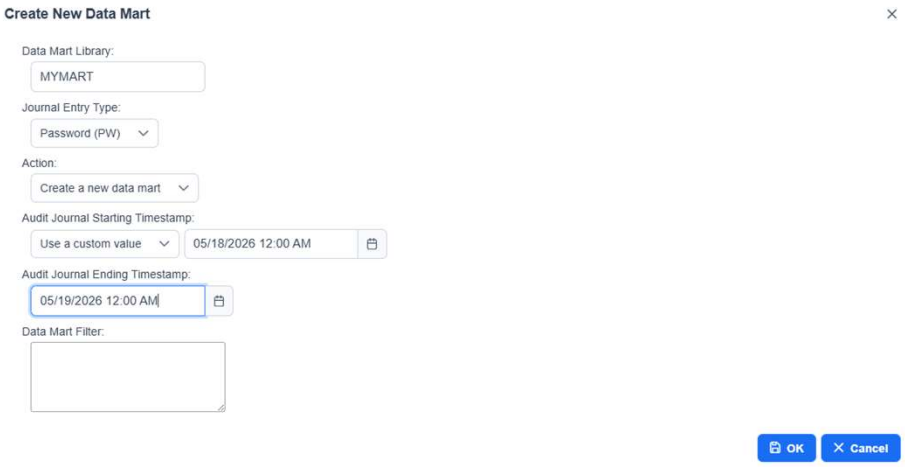
Operationalizing Audit Data



FORTRA

Reading Audit Journal Data: Navigator for i

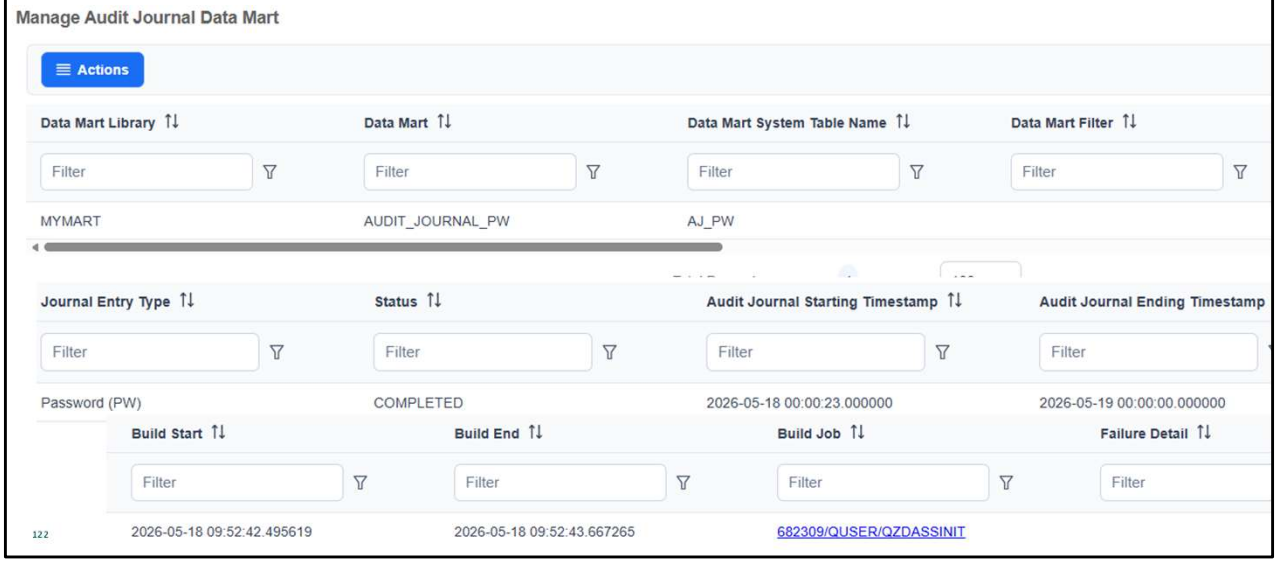
Operationalizing Audit Data



FORTRA

Reading Audit Journal Data: Navigator for i

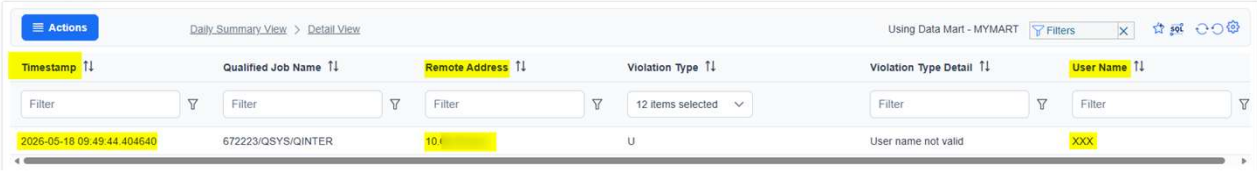
Operationalizing Audit Data



Reading Audit Journal Data: Navigator for i

Operationalizing Audit Data

Password (PW) Detail View



FORTRA

Analyzing Data with SQL

Operationalizing Audit Data

- The amount of data in the security audit journal can be massive
- Production systems log millions of entries each day
- What's a good way to make sense of massive data?
- SQL!
- How to start?

Analyzing Data with SQL

Operationalizing Audit Data

- Data sources: `QSYS2.DISPLAY_JOURNAL()`, `AUDIT_JOURNAL_DATAMART`-generated tables, `SYSTOOLS.AUDIT_JOURNAL_`*`XX`* functions
- Always:
  - Define start time for journal extract
  - Start with a short time, no longer than 10 minutes, to avoid being inundated

FORTRA

*CURRENT means you only get the data from the currently attached receiver, nothing before that
*CURCHAIN you get the data from all receivers that are or were attached to QAUDJRN

Analyzing Data with SQL

Operationalizing Audit Data

- General SQL functionality that is useful to manage mass data like audit journal entries:
  - Limit output: `FETCH FIRST n ROWS ONLY; OPTIMIZE FOR n ROWS`
  - Grouping: `GROUP BY`
  - Sorting: `ORDER BY`
  - Aggregation functions: `COUNT()`, `SUM()`, …
  - Selection: `WHERE x = y`, `WHERE x in (a, b, c)`, `HAVING`

FORTRA

Analyzing Data with SQL: Example

Operationalizing Audit Data

```
with JOURNALENTRIES as (
      select
        jt.current_user as the_current_user, job_name, program_name,
        journal_entry_type CONCAT '-' CONCAT
        cast(cast(substr(entry_data, 1, 1) as char(1)) as char(1) ccsid 37) as entry_type_and_subtype
        from table (
              QSYS2.DISPLAY_JOURNAL(
                JOURNAL_LIBRARY          => ('QSYS'),
                JOURNAL_NAME             => ('QAUDJRN'),
                STARTING_TIMESTAMP       => (CURRENT_TIMESTAMP - 10 minutes),
                JOURNAL_CODES            => 'T',
                INCLUDE_INTERNAL         => 'NO'  -- not interested in journal management entries
                )
           ) as JT)
select count(*) as number_entries, entry_type_and_subtype, job_name, the_current_user, program_name
from JOURNALENTRIES
group by entry_type_and_subtype, job_name, the_current_user, program_name
order by count(*) desc fetch first 30 rows only
```


FORTRA

Subselect useful for readability and can be used to get the total count of the journal entries

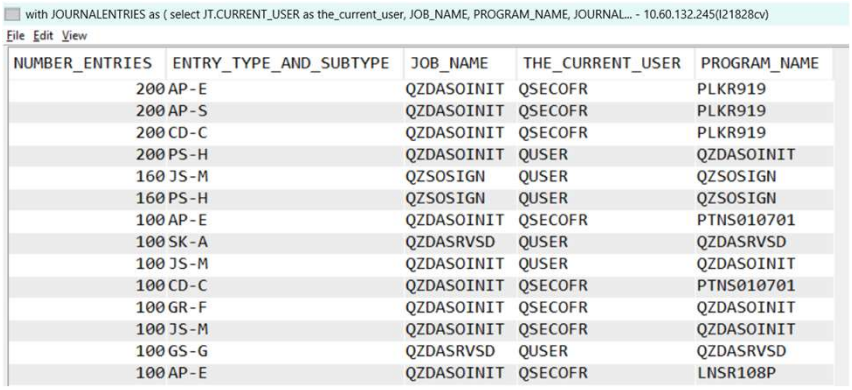
Double cast: needed to convert the subtype to a usable CCSID

DISPLAY_JOURNAL is the main function

Grouping into clusters, sorting, limiting number of output rows

Analyzing Data with SQL: Sample Result

Operationalizing Audit Data


with JOURNALENTRIES as (select JT.CURRENT_USER as the_current_user, JOB_NAME, PROGRAM_NAME, JOURNAL... - 10.60.132.245(I21828cv)

File Edit View

NUMBER_ENTRIES	ENTRY_TYPE_AND_SUBTYPE	JOB_NAME	THE_CURRENT_USER	PROGRAM_NAME
200	AP-E	QZDASOINIT	QSECOFR	PLKR919
200	AP-S	QZDASOINIT	QSECOFR	PLKR919
200	CD-C	QZDASOINIT	QSECOFR	PLKR919
200	PS-H	QZDASOINIT	QUSER	QZDASOINIT
160	JS-M	QZSOSIGN	QUSER	QZSOSIGN
160	PS-H	QZSOSIGN	QUSER	QZSOSIGN
100	AP-E	QZDASOINIT	QSECOFR	PTNS010701
100	SK-A	QZDASRVSD	QUSER	QZDASRVSD
100	JS-M	QZDASOINIT	QUSER	QZDASOINIT
100	CD-C	QZDASOINIT	QSECOFR	PTNS010701
100	GR-F	QZDASOINIT	QSECOFR	QZDASOINIT
100	JS-M	QZDASOINIT	QSECOFR	QZDASOINIT
100	GS-G	QZDASRVSD	QUSER	QZDASRVSD
100	AP-E	QZDASOINIT	QSECOFR	LNSR108P

FORTRA

Here's a sample result from the query you just saw

Analyzing Data with SQL

Operationalizing Audit Data

- Tip: Let AI help you with creating the queries!

Audit Journal Cluster Analysis with subtypes.sql

```
2158                on RTRIM(RE.JOB_NAME) LIKE IJN_USER.JOBNAME and RE.THE_CURRENT_USER = IJN_USER.CURRENTUSER
2159            -- This second join is the fallback, it matches on job name ONLY, to a blank user in the table
2160            left join
2161                INTERPRETEDJOBNAMES IJN_WILD
2162                on RTRIM(RE.JOB_NAME) LIKE IJN_WILD.JOBNAME and IJN_WILD.CURRENTUSER = ''
2163            -- Cross join the necessary CTEs to make their values available.
2164            cross join CONSTANTS C
2165            cross join ANALYSIS_WINDOW AW
2166            cross join TOTALS T
2167        where
2168        -- Filtering:
2169        -- 1. Limit to the top N rows as defined by the C.MAX_ROWS constant.
2170        --    Note: Cannot use "FETCH FIRST n ROWS ONLY", as that only supports literals for the value of "n"
2171            RE.RN <= C.MAX_ROWS
2172        -- 2. Ensure the combination meets the minimum percentage threshold, again, defined in the constants.
2173            and cast((RE.NUMBER_OF_ENTRIES * 100.0) / T.TOTAL_ENTRIES as decimal(5, 2)) >= C.MIN_PERCENTAGE
2174        for read only
2175        -- You can optionally adjust the "optimize for" value below to match the MAX_ROWS constant.
2176        optimize for 50 rows
2177        ;
```

129

FORTRA

I once wanted my SQL query to add the descriptions for the entry types and subtypes. I could have used the SYSTOOLS.AUDIT_JOURNAL functions, but I wanted something with more control.

I gave Claude AI my SQL query and the IBM i Security Reference PDF. I told it to change the query so the output would include the full descriptions for entry types and subtypes. The AI added that functionality two, three tries.

In the future, you will have see agentic AI creating the queries for you based on the questions you have

Analyzing Data with SQL

Operationalizing Audit Data

- These are just some examples of what you can do with SQL.
- Use your imagination

FORTRA

PASE code: Typically code that uses pipes to send output of one command to another command.
JS entries reflect an OS difference — in Linux, creating a new process is cheap, on IBM i, creating a job is expensive, lots of bookkeeping

Seeking Imbalance

Operationalizing Audit Data

- Idea: Under normal operations, QAUDJRN contains a balanced mix of entries
- Look for “lumpy” audit journal data to identify security or operational issues
- Cluster by:
  Entry type + Subtype + Unqualified job name (just job name, w/o job user or job number)
  + current user + program name
- Rule of thumb: No single cluster should be above 15% of all entries
- If there is a cluster at more than 15% ➔ Investigate!

Seeking Imbalance

Operationalizing Audit Data

- Example:

"Number of Entries"	Percentage	Entries per Second	Entry Source	"Entry Type"	Subtype / EPM Server	Interpreted Type and Subtype
1186	5.64	0.329	Security auditing	PS	H	Profile swap: Profile handle ge
1180	5.61	0.327	Security auditing	AP	S	Obtaining adopted authority: St
1180	5.61	0.327	Security auditing	CD	C	Command string audit: Command r
1180	5.61	0.327	Security auditing	AP	E	Obtaining adopted authority: En
956	4.55	0.265	Security auditing	PS	H	Profile swap: Profile handle ge
950	4.52	0.263	Security auditing	JS	M	Job Change: Change profile or g
658	3.13	0.182	Security auditing	CD	C	Command string audit: Command r
591	2.81	0.164	Security auditing	SK	A	Sockets Connections: Accept con

- Typical causes of imbalance:
  - Looping jobs
  - Password changed & automated process uses old password
  - Excessive object access logging (ZC / ZR entries)
- Sometimes imbalance ok:
  - Result of system recovery operations
  - Massive volume of Job-start JS entries ➔ PASE code ported from Linux

132

FORTRA

PASE code: Typically code that uses pipes to send output of one command to another command.
JS entries reflect an OS difference — in Linux, creating a new process is cheap, on IBM i, creating a job is expensive, lots of bookkeeping

Applications

Operationalizing Audit Data

- Commercial applications build on APIs, CL commands, SQL functions for things like:
  - Report creation
  - Remote logging and SIEM/SOAR integration
  - Notification

FORTRA

Applications: Report Creation

Operationalizing Audit Data

- Automatic, customizable report creation:
  - Makes report creation efficient
  - Easy application of filters, adjustment of sorting, output to PDF etc.
  - Supports overview, investigation, and audit use cases

FORTRA

Applications: Report Creation

Operationalizing Audit Data

FORTRA

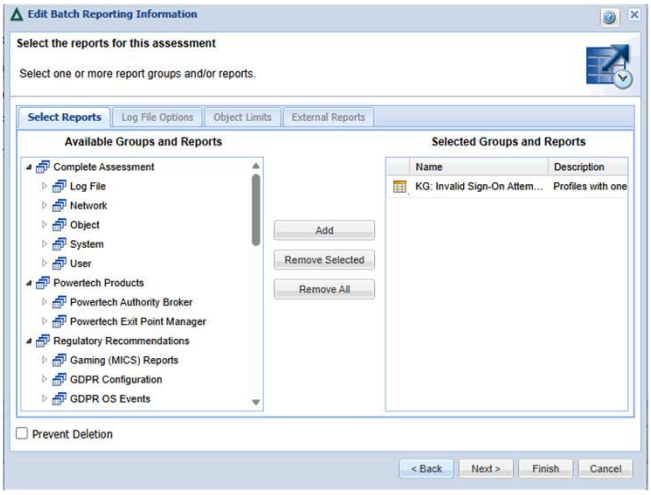
(T:SF) Actions to Spooled Files

Jun 12, 2026
5:51:57 PM

System	Timestamp	Entry Type	File	File Lib	Obj Type	SPLF	Short SPLF Nbr	OUTQ	OUTQ Lib	SPLF Nbr	JE Job Name	JE Job Nbr	JE Job User	JE Current User
HS720P14	06/06/2026 02:01:11.880	Created	Q04079N015	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	DLTAUDRCV	766334	THOMASK	THOMASK
HS720P14	06/06/2026 02:03:12.823	Created	Q04079N017	QSPL	*FILE	PT_TRACE	0001	QEZDEBUG	QUSRSYS	000001	CM0000B	766376	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 02:03:12.924	Created	Q04079N016	QSPL	*FILE	PT_TRACE	0001	QEZDEBUG	QUSRSYS	000001	CM0000B	766375	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 02:03:25.087	Deleted	Q04079N017	QSPL	*FILE	PT_TRACE	0001	QEZDEBUG	QUSRSYS	000001	CM0000B	766376	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 02:03:40.446	Created	Q04079N017	QSPL	*FILE	QPSECPWD	0002	QPRINT	QGPL	000002	CM0000B	766375	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 02:03:41.251	Deleted	Q04079N017	QSPL	*FILE	QPSECPWD	0002	QPRINT	QGPL	000002	CM0000B	766375	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 02:03:41.641	Deleted	Q04079N016	QSPL	*FILE	PT_TRACE	0001	QEZDEBUG	QUSRSYS	000001	CM0000B	766375	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 02:03:41.717	Created	Q04079N017	QSPL	*FILE	PT_TRACE	0003	QEZDEBUG	QUSRSYS	000003	CM0000B	766375	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 02:03:42.182	Deleted	Q04079N017	QSPL	*FILE	PT_TRACE	0003	QEZDEBUG	QUSRSYS	000003	CM0000B	766375	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 04:55:04.419	Created	Q04079N016	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZDASOINIT	766782	QUSER	QUSER
HS720P14	06/06/2026 04:55:04.509	Created	Q04079N017	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZDASOINIT	766832	QUSER	QUSER
HS720P14	06/06/2026 04:55:05.351	Created	Q04079N012	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZDASOINIT	766831	QUSER	QUSER
HS720P14	06/06/2026 04:55:07.520	Created	Q04079N004	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	SUMCAPTRAN	735463	PTUSER	PTUSER
HS720P14	06/06/2026 04:55:08.229	Created	Q04079N006	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZSHSH	735625	FAHUSER	FAHUSER
HS720P14	06/06/2026 04:55:08.785	Created	Q04079N004	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZSHSH	735623	FAHUSER	FAHUSER
HS720P14	06/06/2026 04:55:09.117	Created	Q04079N003	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZSHSH	735622	FAHUSER	FAHUSER
HS720P14	06/06/2026 04:55:09.543	Created	Q04079N005	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZSHSH	735624	FAHUSER	FAHUSER
HS720P14	06/06/2026 04:55:11.540	Created	Q04079N004	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZDASSINIT	735243	QUSER	QUSER
HS720P14	06/06/2026 04:55:11.846	Created	Q04079N011	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZSHSH	735290	STANDGUARD	AVSECOFR
HS720P14	06/06/2026 04:55:12.979	Created	Q04079N004	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZDASSINIT	749723	QUSER	QUSER
HS720P14	06/06/2026 04:55:14.182	Deleted	Q04079N005	QSPL	*FILE	PT_TRACE	0001	QEZDEBUG	QUSRSYS	000001	PLCMMON	735484	PLCMOWN	PLCMOWN
HS720P14	06/06/2026 04:55:15.545	Created	Q04079N005	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZDASOINIT	735433	QUSER	QUSER
HS720P14	06/06/2026 04:55:16.959	Created	Q04079N010	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	AVINSITE	735269	STANDGUARD	AVSECOFR
HS720P14	06/06/2026 04:55:18.922	Created	Q04079N017	QSPL	*FILE	QPJOBLOG	0001	QEZJOBLOG	QUSRSYS	000001	QZDASOINIT	735236	QUSER	QUSER

Applications: Report Creation

Operationalizing Audit Data



FORTRA

Applications: SIEM Integration

Operationalizing Audit Data

- Remote logging:
  - Secure logging: Reduces risks of audit journal data being deleted by an attacker
  - Integrates IBM i with SIEM/SOAR solutions, fixing the “blind spot” for IBM i systems
  - Allows SIEM to cross-correlate events

Applications: Remote Logging

Operationalizing Audit Data

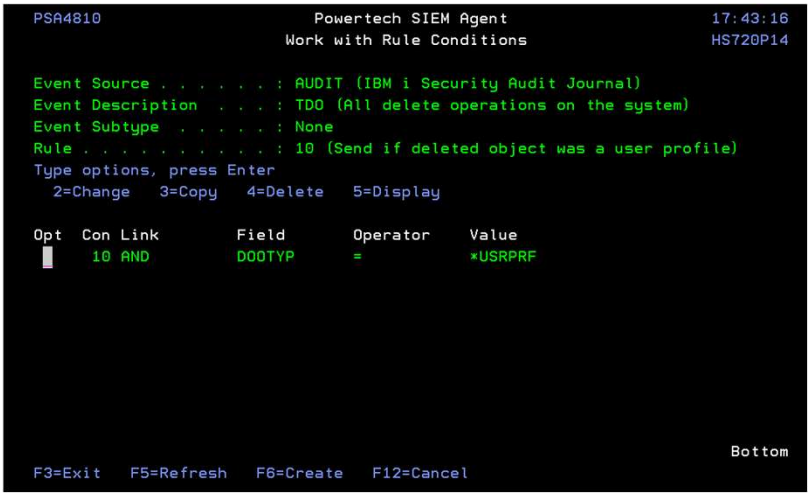
138

FORTRA

Sample application, here you see the mapping of event types

Applications: Remote Logging

Operationalizing Audit Data

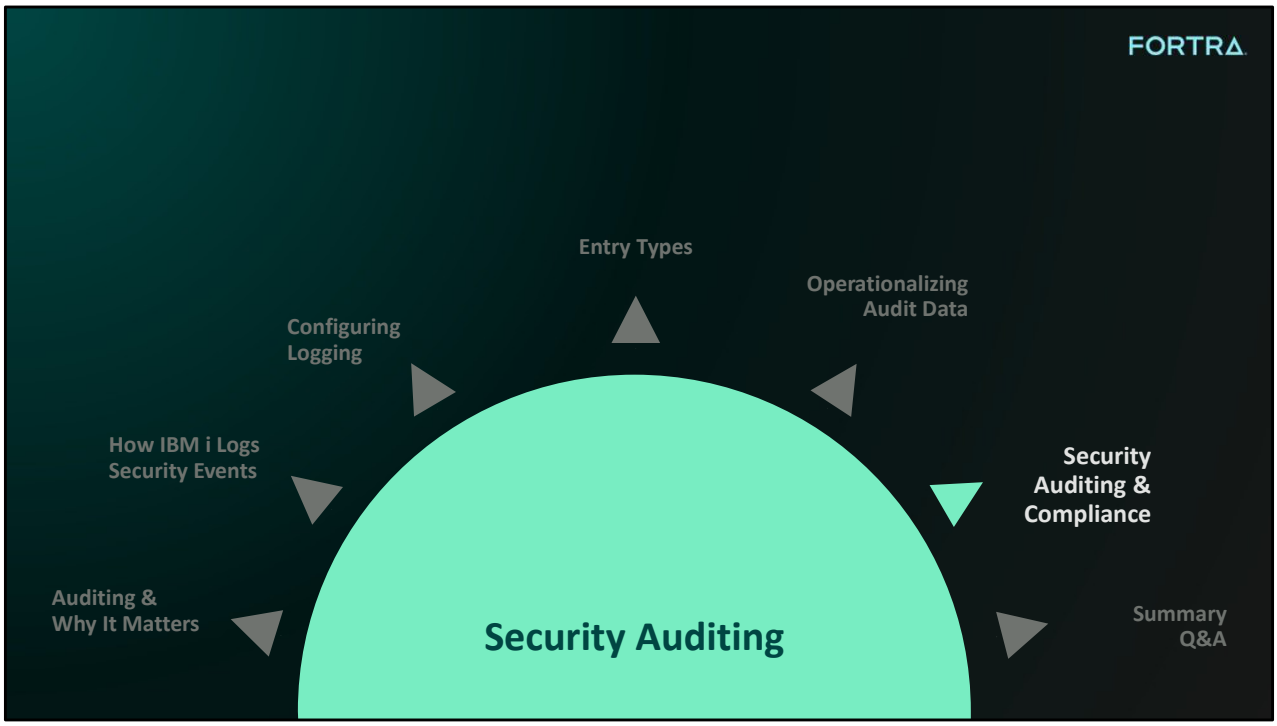


FORTRA

here's an example of a rule condition, "if the deleted object was type *USRPRF". This condition is used in a rule that forwards T:DO events if the deleted object was a user profile.

Summary: Operationalizing Audit Data

- … AKA “Reading Out and Using”
- Required: Audit journal up and running
- Use cases: Overview, Investigation, Auditor, SIEM Integration
- Reading audit journal data
- Analyzing data with SQL
- Applications—examples: SIEM integration


FORTRA
Entry Types
Operationalizing Audit Data
Configuring Logging
How IBM i Logs Security Events
Security Auditing & Compliance
Auditing & Why It Matters
Summary Q&A
Security Auditing

Security Auditing & Compliance

- Compliance frameworks
- Discovering and mapping frameworks with AI
- Protecting auditing data

Frameworks

Security Auditing & Compliance

- A lot of compliance frameworks de facto require security auditing
- Most organizations will fall under at least one of them
- What fkinds of rameworks are there?

FORTRA

Frameworks: Important Aspects

Security Auditing & Compliance

- **Source**
  - Parliaments (GDPR, SOX)
  - Regulatory authority—e.g., national supervisory authority for banking, insurance
  - Service providers (PCI) …
- **Geographic Scope**
  - Supernational (EU …)
  - National (France, Germany)
  - Regional/State (Baden-Württemberg, Alsace …)
- **Sector-Specific**
- **Specific to organizations using a service or type of service (PCI-DSS, SWIFT)**
- **Critical Entity-Specific** (DORA, KRITIS)
- **Source of authority: By law/regulation, organizational self-imposed,** or **business partner-imposed**

FORTRA

organizational self-imposed (“We will follow these rules”), or business partner-imposed (“Do this if you want to do business with us”)

Frameworks: Some Important Compliance Frameworks

Security Auditing & Compliance

- **GDPR**—if personal data is stored on IBM i
  - EU + UK national data protection laws now usually implement GDPR
- **DORA**—financial
- **NIS2**—critical infrastructure
- **PCI-DSS**—if you handle credit card data
- **SWIFT Customer Security Controls Framework**—if you are a bank and work internationally
- **SOX**—if your organization is or wants to be listed on a US stock exchange

Frameworks: When Do They Matter to You?

Security Auditing & Compliance

- … When you are not in the 96% of organizations already using the audit journal
- … When you are thinking about turning logging on additional systems
- … When you want to spend time and effort on tuning the auditing settings and operationalizing audit data
- **In short: When you want to convince management to let you work on or spend money on this topic!**

Using AI for Discovery & Mapping

Security Auditing & Compliance

- Use your favorite AI chatbot (allowed by your organization):
  - **ChatGPT**
  - **Claude**
  - **Perplexity**
  - **…**
- If your AI has a Thinking Mode or Deep Thinking Mode, turn that on first.
- Ask it the following question:

FORTRA

You can translate the question into your own language if you want

Using AI for Discovery & Mapping: Prompt

Security Auditing & Compliance

I want to find compliance frameworks that support enabling the security audit journal on IBM i.

We are <name of your organization>, based out of <city>. We work in the <sector>. <We are a Critical Entity/Designated Entity under {NIS2, national critical infrastructure law}. <We process personally identifiable information on the IBM i.> <We process credit card data on the IBM i.>

We are at least subject to the following frameworks: <... frameworks you already know>.

Identify all relevant regulatory, industry-specific and other compliance or security frameworks that directly or indirectly require, or support, logging security events on IBM i. Check if GDPR applies. For each relevant framework, identify any authoritative guidance documents. For GDPR, guidance documents can come from EDBP and ENISA as well as from national authorities. Ignore the Cyber Resilience Act.

Output a list of the frameworks, explain for each why it applies to my organization. List the clauses in each framework or guidance document that require the logging of security events, and explain how that maps to using the security audit journal on IBM i.

Bring receipts. Do not map to IBM i settings or commands.

148

FORTRA

- Prime the AI by specifying your goal first. This was proven in a scientific experiment to improve quality of answers. General guideline: Don't make the A_I guess what you want.
- Info about your organization helps the AI find the relevant frameworks.
- Guidance documents are very useful, especially if the framework itself is very abstract, like the GDPR. I mention specific guidance sources for GDPR.
- The Cyber Resilience Act is about making tools that you sell to others. You don't sell custom IBM i systems, so that's not relevant.
- "Bring receipts" makes it double-check sources.
- We don't want it to provide us with IBM i specific settings or commands—those suggestions are usually wrong.

Using AI for Discovery & Mapping: Working with Results

Security Auditing & Compliance

- Spot-check the output
- You can always ask follow-up questions: “What does this mean?”, “Are you sure that … ?”
- If you have paid AI, you can ask it to output to a Word document
- Careful when asking about requirement for *specific* QAUDLVL/2 settings, AI tends to overcomply
  - “Of course you need <obscure setting with no practical value>”

FORTRA

Protecting Auditing Data

Security Auditing & Compliance

- Many compliance frameworks require you to use security auditing, but …
- compliance works the other way around, too
- Data in the security audit journal falls under the protection of most compliance frameworks

FORTRA

QAUDJRN must always be *PUBLIC *EXCLUDE
Do not hand out *ALLOBJ like candy

Protecting Auditing Data

Security Auditing & Compliance

- Example: Anything in QAUDJRN that allows a specific user to be identified is “personal data” under GDPR
  - You need to protect it against illegitimate access
  - Admins or compliance reviewing logs for security or system admin purposes is fine
  - Just make sure that Joe and Jane Normal User cannot see the data
- Generally speaking, the **audit journal and data extracted from it must be protected in a way that complies with all relevant compliance frameworks**

QAUDJRN must always be *PUBLIC *EXCLUDE
Do not hand out *ALLOBJ like candy

Summary: Security Auditing & Compliance

- Compliance Frameworks: Dimensions and examples
- Discovering and mapping frameworks with AI
- Protecting auditing data


FORTRA
Summary, Questions & Answers
153

Summary

- Auditing & why it matters
- How IBM i logs security events
- Configuring logging
- Entry types
- Operationalizing audit data
- Security auditing and compliance

FORTRA

Questions? Answers!

155

FORTRA

Bonus Section

156

QAUDLVL2 Recommendations (1/2)

Configuring Logging

Setting	What It Logs	Why Log It?
*SEC…	Changed to auditing settings Anything except for: *SECURITY and *SECIPC	Logs most security-relevant configuration changes. *SECIPC logs interprocess communication, which is rarely useful info; *SECURITY contains *SECIPC
*CREATE	Object creation, except for profile creation	Basic cyber hygiene; your HA software will add that setting anyways.
*DELETE	Object deletion	Basic cyber hygiene; also logs deletion of user profiles. And your HA software will add it.
*AUTFAIL	Authority failures	Just do it
*NETSMBSVR *(from IBM i 7.6)*	Changes to NetServer shares, failed anonymous connections	For your NetServer security woes
*PGMFAIL	System integrity violations	Just do it

QAUDLVL2 Recommendations (2/2)

Configuring Logging

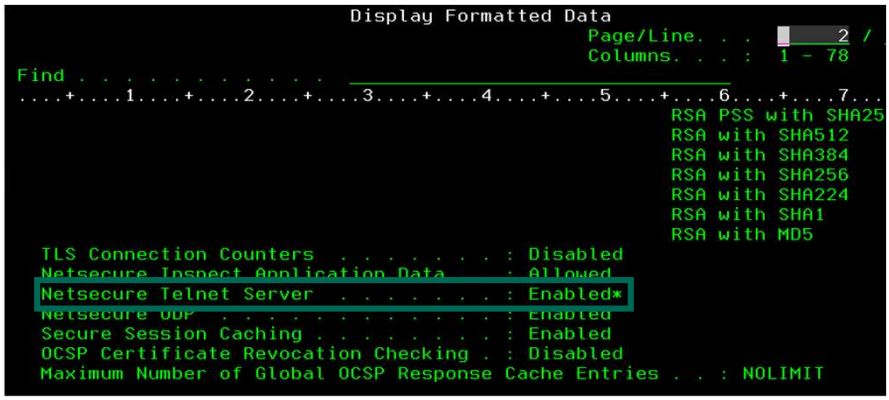
Setting Name	What It Logs	Why Log It?
*SYSMGT	Various systems management changes. If Db2 Mirror is used, will log Db2 Mirror activity.	Changes that will affect system behavior need an audit trail.
*NETFAIL	Failed network activity	Track network failures; useful for forensic analysis
*OBJMGT	Object moves and renames	Basic cyber hygiene.
*NETTELSVR	5250 connections	Basic cyber hygiene
*SERVICE	Service Tools activity (STRSST, DMPOBJ)	Must-have, use of service tools always needs a business justification
*JOBDTA	Job starts, stops, change of current user	Basic cyber hygiene; start of important jobs
*PTFOPR	PTF application and removal	PTF's can impact system behavior, so needs audit trail
*SAVRST	Restore operations	Could be restoring software!

Note: In ST = Service Tool Usage entries, you sometimes see that a DMPOBJ was run. That is usually the OS collecting info after a program crashed. You can ignore that.
*NETTELSVR–you need to enable the logging of Telnet over TLS
*SAVRST—sadly, only logs object restores, not object saves, although object saves could be used for data exfiltration and profiles with *SAVSYS can save any object

QAUDLVL2 Recommendations (3/2 ☺)

Configuring Logging

Additionally, use the TLSCONFIG macro in SST to ensure that log entries are created for 5250 sessions even if the sessions are secured with TLS:



FORTRA

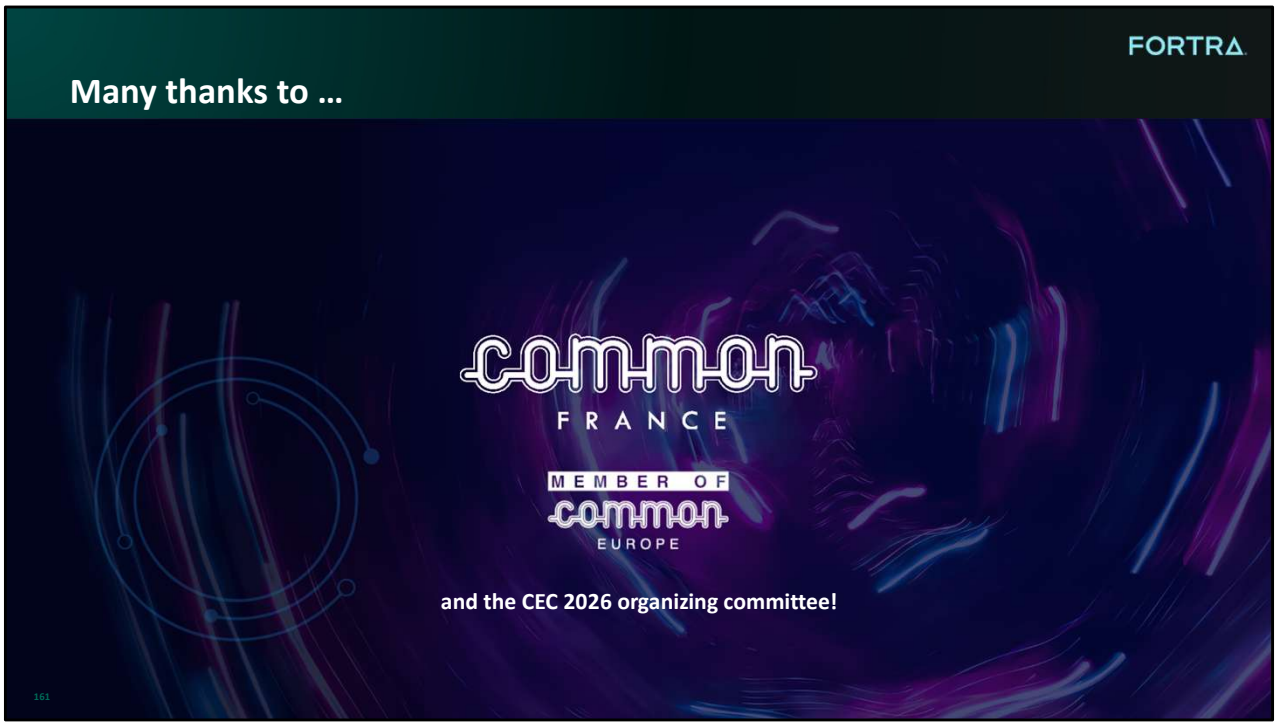
FORTRA

Fortra at CEC Lyon

Meet us at the Fortra stand in the main hall

… Join Fortra's Mike Davison presentations, like **Ready for that next step? Why the hell do you want to be in IT management?** on Tuesday, 10:30, in room Rhône 4

160



Fortra is proud to be a Gold sponsor of CEC Lyon 2026!

Best-of-Breed IBM i Security Products

Perimeter Access Control
Exit Point Manager for IBM i

Native Virus Protection
Antivirus for IBM i, AIX, Linux, Linux on Z

Privileged Access Management
Authority Broker for IBM i

Multi-Factor Authentication
Multi-Factor Authentication for IBM i
RSA SecurID Agent for IBM i

Native Encryption
Encryption for IBM i

Command Monitoring
Command Security for IBM i

Secure Managed File Transfer
GoAnywhere

Free Security Snapshot
Powertech Security Scan

User Provisioning
Identity Manager for IBM i

Self-Service Password Reset
Password Self Help for IBM i

Database Monitoring
Database Monitor for IBM i

Security Information and Event Management
SIEM Agent for IBM i
Event Manager

Compliance Reporting
Compliance Monitor for IBM i

InfoSec Policy Control
Policy Minder for IBM i

Automated Risk Audit
Risk Assessor for IBM i

FORTRA


FORTRA
THANK YOU
power.fortra.com/
kurt.thomas@fortra.com
163