TAXENTRIQ

ENTERPRISE TAX INTELLIGENCE

The Determination Defect Catalogue

Fifteen ways SAP tax determination fails without raising an error — what happens, why testing misses it, and how to check each one.

WHY THIS EXISTS

Determination does not fall over.

When a posting fails you get an error, someone raises a ticket, and it is resolved that week. Determination does not do that. It picks something. The document saves, the invoice goes out, the customer pays it, and nobody knows until someone in the filing team is looking at a box on a return that should have a number in it and does not.

Every defect in this catalogue produces a document that looks completely normal. That is the only thing they have in common, and it is the reason they survive in production for years.

The second thing worth saying up front: not one of these was caused by somebody doing their job badly. The AP clerk who overrode the code was right. The developer who wrote the user exit was solving a real problem. Whoever put an end date on that condition record thought they were being careful. An engine that applies standard rate when nothing matches is correct design.

Every one of them lives in a gap between two people who were both doing their job properly. That is not a competence problem and it cannot be fixed by trying harder. It is a scope problem, and it stays there until somebody's name is against the space in between.

HOW TO USE IT

Each entry has a check that takes between ten minutes and an hour. They are written so that a functional consultant, a tax manager or a delivery lead can run them without specialist tooling. The consolidated list of all fifteen checks is on the final page.

DETERMINATION DEFECT #01

The silent default

A missing tax classification does not fail. It falls through to a rate that looks plausible.

Customer tax classification	1
Material tax classification	–
Access sequence	FALLS THROUGH
Treatment applied	A1 standard rate
Treatment expected	E1 exempt
Error raised	NONE

WHAT HAPPENED

The material tax classification was blank. The access sequence looked for the specific combination, did not find it, and fell through to a more general condition record. A rate was returned. It was the wrong rate, and nothing about the transaction says so.

WHY TESTING MISSES IT

A missing classification is a master data gap, not a configuration gap. The scenario passes design review because the design is correct, and passes unit test because the test data was created properly. It reaches production on materials extended in a hurry — usually the ones added after the test data was built.

> **HOW TO CHECK**
>
> Pull the tax classification field across your material master for the countries you are live in and count the blanks. Then check which condition record those materials would land on. If the answer is a standard-rate fallback and any should be exempt or reduced, you have posted documents to correct.

DETERMINATION DEFECT #02

Determined. Then lost.

Determination got it right. The account key sent it to a generic account, and the box on the return stayed empty.

Determination result	E1 reverse charge
Account key	MWS generic output
G/L account posted	175000 output VAT
Reverse charge box on return	0.00
Posted vs filed	RECONCILES

WHAT HAPPENED

Determination correctly identified a reverse charge supply. The account key then routed it to the generic output VAT account alongside ordinary domestic sales, and return preparation had no way to tell the two apart. The value sat in the wrong box, and the ledger agrees with it.

WHY TESTING MISSES IT

Determination testing stops at the determination result. The scenario passed — expected E1, actual E1. Nobody carried the transaction through the account key, into the G/L and out onto the return, because that path crosses three teams and no single test script owns all of it.

> **HOW TO CHECK**
>
> Take one reverse charge transaction from last period and follow it to the account it posted to. Then find that value on the return. If you cannot point at the box, neither can an auditor.

DETERMINATION DEFECT #03

Determined twice

Determination ran at the order and again at the invoice. The master data moved in between.

Sales order · 12 Mar	**A1 standard rate**
Customer tax classification	**CHANGED 04 Apr**
Billing document · 09 May	**E1 exempt**
Order vs invoice treatment	**DISAGREE**
Configuration changed	**NONE**

WHAT HAPPENED

Determination ran twice. Between the two runs somebody maintained the customer master — an exemption certificate loaded, a VAT ID validated, a data cleanse tidying classifications. Ordinary maintenance, done properly. The invoice re-determined against the new picture and returned a different answer. Nothing was transported.

WHY TESTING MISSES IT

Test cycles happen inside an afternoon. You create the data, run the flow, check the result, and nothing moves in between. There is no test script for waiting six weeks while someone in another team changes a field — and that change goes through master data governance, not tax.

> ### HOW TO CHECK
>
> Pick a month and find billing documents where the tax code does not match the tax code on the sales order behind them. Any count above zero needs an explanation. Partial billing compounds it: one order, two invoices either side of the change, two defensible treatments.

DETERMINATION DEFECT #04

Taxed by default

Nothing matched in the engine, so it fell back to standard rate. That answer hides in the biggest population you have.

Engine call	HTTP 200 · success
Rule matched in engine	NONE
Engine default applied	STANDARD RATE
Tax charged to customer	19.00 %
Treatment expected	reverse charge
Document status	COMPLETE

WHAT HAPPENED

Nothing matched. The product mapping was absent, or the element carrying the classification arrived empty, or the transaction shape hit no rule anyone had written. The engine applied its default and returned a number. That default is deliberate and correct design — under-collecting is the worse exposure.

WHY TESTING MISSES IT

Standard rate is the expected result for most scenarios, so it never looks anomalous. Nobody asks why the engine returned what it returned; the assertion is on the number, not the reason. And unlike a wrongly-zero line, a wrongly-standard line sits inside your largest population.

> ### HOW TO CHECK
>
> Your engine writes which rule fired into its own audit output. Pull a period and filter for lines where no rule matched and the default was applied. It is the fastest read available on whether your mapping is complete.

DETERMINATION DEFECT #05

Someone fixed it

A user changed the tax code by hand. The document is right. Determination is still wrong, and nobody will report it.

Determination returned	**A1 standard rate**
User override on document	**E1 reverse charge**
Document as posted	**CORRECT**
Defect raised	**NONE**
Root cause fixed	**NO**
Next month	**OVERRIDE AGAIN**

WHAT HAPPENED

Determination returned the wrong treatment on a vendor invoice. Somebody in AP noticed and changed the code. They were right to — the invoice had to go out and they knew the answer. It becomes how that vendor is processed: undocumented, in one person's head, invisible to anyone measuring determination accuracy.

WHY TESTING MISSES IT

Testing did not miss it. The config is genuinely wrong and a proper scenario set would have caught it pre-go-live. What hides it is what happens afterwards: there is no test cycle running in month fourteen, and the person absorbing the error every month is why nobody knows it is there.

> **HOW TO CHECK**
>
> Do not run a report. Ask the AP team how often they change a tax code by hand and on which vendors. They will tell you immediately and in detail, because it annoys them. Nobody has ever asked them.

DETERMINATION DEFECT #06

Not in the config

Every setting is correct. The answer is being changed after determination runs, by code nobody remembers writing.

Condition records	CORRECT
Access sequence	CORRECT
Engine mapping	CORRECT
Tax code determined	E1 reverse charge
User exit · MV45AFZZ	OVERWRITES TKOMK
Tax code posted	A1 standard rate

WHAT HAPPENED

An enhancement. Somebody needed a field populated for one scenario — a plant, a document type, one awkward customer — with three weeks to go-live, so it went into a user exit. It was correct for that scenario. The scenario changed; the code did not. Nobody documented it.

WHY TESTING MISSES IT

Everyone looks in config, and no symptom points anywhere else. A wrong tax code says check the condition records, which is exactly where the answer is not. A determination review that only reads configuration is incomplete: config tells you what should happen, not what does.

> ### HOW TO CHECK
>
> Pull the active enhancement implementations touching pricing and the tax interface. For each, find someone who can say why it is there. Anything nobody can account for is your finding. If config says one thing and the document says another, the difference is code — you need a runtime trace.

DETERMINATION DEFECT #07

The wrong date

SAP holds several candidate dates. The connector maps one. It is correct until the day something changes.

Service rendered	**28 Dec**
Invoice posted	**14 Jan**
Date passed to engine	**POSTING DATE**
Rate version applied	**current year · 21.0 %**
Rate in force at supply	**prior year · 19.0 %**
Document status	**COMPLETE**

WHAT HAPPENED

The engine needs a date to decide which rate version applies, which registration was valid, which rule was in force. SAP offers document date, posting date, pricing date, service rendered date and more. The connector maps one. Posting date is rarely correct and frequently chosen, because it is never null.

WHY TESTING MISSES IT

In a test system every date is today. Order, delivery, invoice and posting all carry the same value, so a date-mapping error is structurally undetectable. Your test cycle cannot fail this scenario — not did not, cannot. A connector upgrade or engine migration can also change the mapping silently.

> ### HOW TO CHECK
>
> Create one document where the dates deliberately differ — a service date in a prior period, posted today. Read the date the engine actually determined on from its audit output. If it is not the date you would defend to a tax authority, you have a finding and probably a history.

DETERMINATION DEFECT #08

It never asked

The document posted with a tax code that came from somewhere else. Determination was never involved.

Entry channel	**EDI inbound · IDoc**
Determination called	**NO**
Tax code source	**INBOUND MAPPING TABLE**
Table last maintained	**2019**
Tax code posted	**A1 standard rate**
Document status	**COMPLETE**

WHAT HAPPENED

It came in through a different door — inbound EDI, a procurement platform, a billing bolt-on, an interface built for one business unit. That channel was configured separately and the tax code came from a mapping table maintained during the original implementation and untouched since.

WHY TESTING MISSES IT

Testing goes through the front door. Interfaces do get tested, but for whether the document posts; the tax code is incidental and whoever writes that script is not a tax person. The largest uncounted population is usually cutover itself — migrated open items carry legacy tax codes that never touched determination.

> ### HOW TO CHECK
>
> Take one period and group tax-relevant line items by what created them. Every entry channel should be nameable and someone should be able to say where the tax code comes from in each. The finding is usually not a bad channel — it is that nobody has ever had the list.

DETERMINATION DEFECT #09

Valid to yesterday

Nobody changed anything. A validity date passed, the access fell through, and a different tax code started posting.

Condition record	EXISTS
Valid from	01.01.2019
Valid to	31.12.2025
Document date	02.01.2026
Access result	FALLS THROUGH
Tax code posted	A1 standard rate

WHAT HAPPENED

A condition record carried an end date set during the original build, probably as an intended review trigger. Nobody reviewed it. At midnight the record stopped being found, the access moved to a more general one, and the answer changed. The system did exactly what it was configured to do.

WHY TESTING MISSES IT

Testing happened when the record was valid, and no test reruns itself years later. This is configuration with a time dimension, and nothing in a normal change process covers a system behaving differently on a future date without anyone doing anything. It degrades rather than fails — only the covered combinations move.

> ### HOW TO CHECK
>
> List tax condition records sorted by validity end date, ascending. Anything expiring in the next twelve months needs an owner and a decision. For most tax condition records the open-ended date is the safer choice, with a scheduled review as the control rather than a cliff.

DETERMINATION DEFECT #10

Two lists

SAP knows where you are registered. So does the engine. Nothing reconciles the two.

Registrations in SAP	14
Registrations in the engine	12
Missing	NL PL
NL domestic supply seen as	CROSS-BORDER
VAT charged	0.00
VAT actually due	local rate

WHAT HAPPENED

A registration was maintained properly in SAP — company code data, registration number, plant — and never maintained in the engine. The engine still believes there is no establishment there, so a domestic supply looks like a cross-border one. It returns a correct treatment for a company that is not registered. You are.

WHY TESTING MISSES IT

Registrations are set up once, at the start, and tested then. New ones arrive afterwards as business events — a warehouse opens, an acquisition completes, a threshold is crossed. Each goes through a tax process and a master data process, and neither has a step called update the engine.

> ### HOW TO CHECK
>
> Print both lists and put them side by side. Ten minutes. The reverse matters too: a deregistration updated in SAP and not the engine means you keep charging local VAT under a registration you no longer hold.

DETERMINATION DEFECT #11

Nothing to tax

The rate was right. There was no base for it to apply to.

Item category	**free of charge**
Net value	**0.00**
Tax base	**0.00**
Rate determined	**CORRECT**
VAT calculated	**0.00**
VAT due on cost	**local rate**

WHAT HAPPENED

Goods left the business — a sample, a warranty replacement, a goodwill shipment. The item carried a free-of-charge category so the net value was zero. Determination applied the correct rate to a zero base and returned zero. In many jurisdictions that is still a supply, with VAT due on cost or open market value.

WHY TESTING MISSES IT

Free-of-charge flows are not in the scenario set because they are not commercial transactions. Nobody blueprints giving things away. They arrive through marketing, quality, customer service and HR — functions with no reason to speak to tax. Everything else in determination is about which treatment; this is about what tax is calculated on.

> ### HOW TO CHECK
>
> Pull zero-value goods movements and deliveries for one period. Ask which actually left the business, then whether any VAT was accounted for. If the answer is that the tax team posts an adjustment, that is defect 05 wearing different clothes.

DETERMINATION DEFECT #12

Recovered anyway

Determination answered what tax applies to the supply. Nobody asked how much of it you can recover.

Supply	**business entertainment**
Rate determined	**CORRECT**
Tax code used	**V1 fully deductible**
Non-deductible portion	**0 %**
Input VAT recovered	**100 %**
Recoverable in law	**0 %**

WHAT HAPPENED

Deductibility is not a property of the supply. It is a property of you — what you bought it for, what your business does, what your recovery position is this year. In SAP that lives in the tax code's non-deductible portion. If everything lands on a fully deductible code, you recover everything.

WHY TESTING MISSES IT

Scenario sets assert on the rate and the tax code. Nobody writes an expected result saying none of this should be recoverable. Deductibility is treated as a tax team topic rather than a determination topic, so it never enters test scope — and the tax team assumes the codes reflect it.

> ### HOW TO CHECK
>
> List the AP tax codes in use with their non-deductible portions. If every one is zero, you are either fully taxable with no blocked categories anywhere in your spend, which is rare, or nobody built it. For partially exempt businesses the recovery percentage moves annually; the codes usually do not.

DETERMINATION DEFECT #13

It didn't net to zero

A reversal should mirror the document it reverses. Determination has no concept of mirroring.

Original invoice · Nov	A1 19.0 %
Credit memo · Feb	A1 21.0 %
Reference to invoice	NONE
Determination re-run	YES
Revenue netted	0.00
VAT residue	2.0 % of base

WHAT HAPPENED

The credit memo was created without a reference to the original invoice, or with one that does not carry the tax treatment. Determination ran again against today's master data and today's rates. A rate change, a moved classification, a new registration or an expired certificate all produce the same divergence.

WHY TESTING MISSES IT

Credit memos do get tested — on the same afternoon as the invoice, against the same master data, with the same rates in force. The test cannot produce a divergence because nothing has diverged yet. The same structural blind spot as defect 07.

> ### HOW TO CHECK
>
> Pull credit memos for a period and compare tax code and rate against the invoice they reference. Start with the ones that have no reference at all. It surfaces in the VAT account as a residue that belongs to nobody, not in revenue, which nets cleanly.

DETERMINATION DEFECT #14

Valid when you checked

The number was validated once. Nobody re-validated it, and nobody tells you when it stops being valid.

Customer VAT ID	ON FILE
Validated at creation	2019
Re-validated since	NEVER
Status today	INVALID
Supply zero-rated as	intra-community
VAT now due	local rate

WHAT HAPPENED

The number was valid when the customer was created. Since then they deregistered, restructured, or their number changed. There is no feed and no alert. The number sits in master data looking exactly as valid as the day it was checked, and the exemption generally depends on it being valid at the time of supply.

WHY TESTING MISSES IT

Validation is tested as a process — can we validate, does it store, does it drive determination. All of that works. What cannot be tested is the passage of time on data owned by somebody else. Every other defect here lives in your landscape; this one lives in someone else's.

> ### HOW TO CHECK
>
> Re-run validation across active customers with intra-community supplies in the last twelve months and compare against what is on file. The fix is not a clean-up project. The data goes stale continuously, so the control is scheduled re-validation.

DETERMINATION DEFECT #15

Neither side was wrong

Two company codes, one transaction, two determinations. Both correct on their own terms.

Selling entity determined	**E1 reverse charge**
Receiving entity determined	**A1 standard rate**
Same transaction	**YES**
Reconciles at group level	**NO**
Input VAT claimed one side	**YES**
Output VAT declared other	**NO**

WHAT HAPPENED

One transaction determined twice by two company codes that do not share a view of it. The selling side sees a customer; the receiving side sees a vendor. Neither knows it is looking at half of an intercompany flow, and nothing in either document says the other leg was decided elsewhere.

WHY TESTING MISSES IT

Scenarios are built and tested per company code. That is the natural unit — it is how config is organised and how workstreams are split. Testing both legs together requires holding two company codes in view at once, and the entities are usually owned by different people, each correctly testing their own scope.

> **HOW TO CHECK**
>
> Take intercompany flows for a period and compare the treatment on each leg. Not the amounts, which usually tie — the tax codes. Any pair that does not correspond needs an explanation.

THE FIFTEEN CHECKS

One page, if you only keep one.

01	**The silent default**	Count blank material tax classifications, then see which condition record they fall through to.
02	**Determined. Then lost.**	Follow one reverse charge posting to its G/L account, then find that value on the return.
03	**Determined twice**	Compare billing document tax codes against the sales orders behind them.
04	**Taxed by default**	Filter the engine audit output for lines where no rule matched and the default applied.
05	**Someone fixed it**	Ask the AP team how often they change a tax code by hand, and on which vendors.
06	**Not in the config**	List active enhancements touching pricing and the tax interface. Account for every one.
07	**The wrong date**	Post a document with a prior-period service date and read the date the engine used.
08	**It never asked**	Group tax-relevant line items by what created them. Every entry channel should be nameable.
09	**Valid to yesterday**	Sort tax condition records by validity end date. Anything expiring within a year needs an owner.
10	**Two lists**	Print your registration list from SAP and from the engine. Put them side by side.
11	**Nothing to tax**	Pull zero-value deliveries. Ask which left the business and whether any VAT was accounted for.
12	**Recovered anyway**	List AP tax codes with their non-deductible portions. Every one at zero is a finding.
13	**It didn't net to zero**	Compare credit memo tax codes against the invoices they reference. Start with unreferenced ones.
14	**Valid when you checked**	Re-run VAT ID validation across customers with intra-community supplies in the last year.
15	**Neither side was wrong**	Compare the tax code on each leg of your intercompany flows. The codes, not the amounts.

TAXENTRIQ

The most useful first conversation is usually about your last cutover — who caught the tax defects, when, and what it cost to fix.

We implement, test and support tax determination in SAP across ONESOURCE, Vertex and native SAP configuration.

info@taxentriq.ai · taxentriq.ai