Fiche pratique

Débogage SAP pour consultants fonctionnels

🎯 Objectif

Apprendre à identifier, analyser et corriger les anomalies dans SAP sans être développeur, en utilisant les outils standards de débogage et de traçage.

🧭 1. Principes de base

🎯 Quand utiliser le débogage ?

- Quand un traitement s'interrompt sans explication.
- Quand un message d'erreur n'indique pas clairement la cause.
- Quand un job, un IDoc ou une sortie (BRFplus/NAST) échoue sans trace fonctionnelle.

⚙️ Comment activer le mode debug ?

- **/h** → mode debug classique (avant validation d'un écran).
- **/ha** → debug "tous modules", même sans écran.
- **/hs** → inclut le code système.
- **SM37 → JDBG** → debug d'un job de fond.
- **BREAK-POINT** → inséré directement dans le code (utile côté technique).

🔬 2. Méthodes d'analyse

🧩 a. Approche préventive

Avant de tester :

- Lire la logique (doc, logs, règles BRFplus).
- Vérifier variables non initialisées, paramètres incohérents, données manquantes.
- Contrôler les conditions et sélections d'entrée.

🔁 b. Reproduire le bug

Toujours reproduire dans un contexte maîtrisé. **4 types de bugs typiques :**

Type de bug	Méthode d'analyse
Erreur avec message	Lancer le debug sur la transaction et lire le message (SE91, point d'arrêt sur message).
Blocage sans message	Utiliser SM50, SM21 ou ST22 pour tracer.
Données incohérentes	Observer les variables dans le débogueur.
Erreur de rendu ou affichage	Vérifier les routines d'affichage et structures de sortie.

🔍 c. Approche "fonctionnelle + technique"

- **Fonctionnelle** : comprendre le process métier, le customizing, et les tables d'entrée.
- **Technique** : suivre le code pas à pas pour confirmer la logique.

👉 La combinaison des deux fait la différence entre un "suiveur" et un "analyste".

⚡ 3. Outils clés

Outil	Usage principal	Exemple concret
ST22	Dumps ABAP	Identifier un programme ou un objet en erreur
SM13	Erreurs de mise à jour	Vérifier les MAJ bloquées (UPDATE_FAILED)
SM37 / JDBG	Jobs en erreur	Debug d'un job échoué en batch
ST01	Traces d'autorisation ou SQL	Vérifier les accès manquants ou les requêtes
SM50	Processus actifs	Identifier blocages / verrous
SLG1	Logs d'application	Messages BRFplus, Output Management, etc.
SE37 / SE80	Debug d'un module fonction	Breakpoint sur un FM précis (ex. APOC_OUTPUT_ISSUE)

🧠 4. Cas pratiques par domaine

🧾 A. Déboguer une sortie BRFplus (S/4HANA)

1. Placer un **breakpoint** dans le FM APOC_OR_ISSUE_OUTPUT.
2. Exécuter le scénario (ex : création commande, livraison...).
3. Si le breakpoint **n'est pas atteint** :
    - Vérifier le **bgRFC Monitor (SBGRFCMON)**.
    - Contrôler les logs dans **SLG1** → objet OUTPUT_CONTROL ou ARIBA_INTEGRATION.
    - Consulter le statut du document : "To be output" / "Pending".
4. Si la règle BRFplus est incorrecte, utiliser **Transaction BRF+** pour simuler.

📦 B. Déboguer un IDoc

1. Transaction WE19 → rejouer un IDoc en mode test.
2. Lancer le **debug** au moment du traitement inbound/outbound.
3. Contrôler la structure (segment, statut, FM de traitement).
4. Si blocage ALE → vérifier les ports et les partenaires (WE20).

⚙️ C. Déboguer un job

1. Transaction SM37 → afficher le job.
2. Sélectionner → JDBG.
3. Le job se relance en debug, à l'endroit exact du plantage.

💾 D. Déboguer un process utilisateur

1. Lancer le scénario depuis la transaction utilisateur.
2. Mettre /h avant la validation critique.
3. Lire pas à pas les valeurs dans le débogueur :
    - Variables locales/globales.
    - Champs internes (structures, tables).
    - Événements conditionnels (IF, LOOP, CASE).

🔧 5. Bonnes pratiques

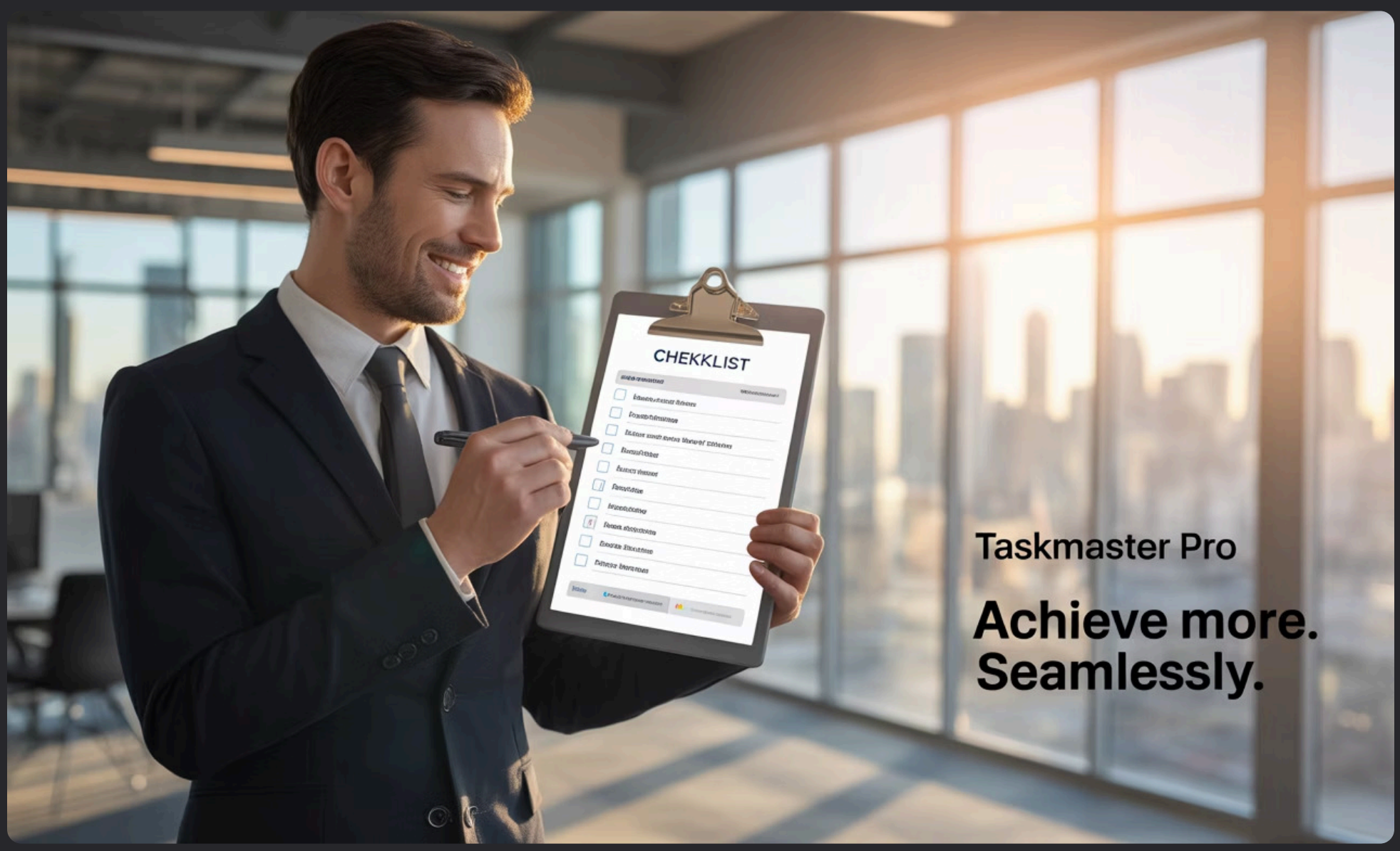
- Toujours documenter le contexte exact du test (utilisateur, date, données).

- Identifier si le problème est **récurrent** ou **isolé**.

- Comparer avec un **cas OK** (approche différentielle).

- Tester en environnement QA avant de modifier un paramétrage.

- Informer le développeur avec **preuve, code, et contexte** (table, FM, message).



📘 6. En résumé

Étape	Objectif	Outil principal
Reproduire l'erreur	Comprendre le contexte	TCode / logs
Lancer le debug	Localiser la cause	/h, JDBG, WE19
Analyser	Suivre les variables, FM, logiques	Debugger
Vérifier les logs	Traces et erreurs applicatives	SLG1, ST22, SM13
Corriger / Escalader	Rédiger un rapport clair au développeur	Word / Notion